MASTER EN SCIENCES INFORMATIQUES

Gestion informatisée de dossiers médicaux au sein d'un hôpital

Goguin, Jean-François

Award date:
1987

Awarding institution:
Universite de Namur

FACULTÉS
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Gestion Informatisée de Dossiers Médicaux au sein d'un hôpital

Jean-François GOGUIN

Mémoire présenté en vue de
l'obtention du diplôme de
Licence en Informatique.

Promoteur : Prof. J. FICHEFET

Chef de projet : Mr. Claude HENRY

ANNEE : 1986-1987

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

Je voudrais remercier,

Mon promoteur J. Fichefet et son assistant J.P. Leclerq et tous les professeurs des Facultés Notre-Dame de la Paix à Namur qui m'ont enseigné l'informatique au cours de ces années d'études.

Mr. C. Henry et tout le service informatique de l'Hôpital St Joseph de Gilly, qui m'ont fourni de précieux conseils.

Les médecins rencontrés qui m'ont initié au monde des dossiers médicaux, tout particulièrement les Dr Nackers et Ratier.

Et enfin, mes parents et amis qui m'ont encouragé.

PLAN

Introduction

1 L'idée du mémoire 1
2 Vocabulaire 1
3 Profil du mémoire 3

Chapitre I : Considérations et Avertissements

1 De la nécessité d'une informatisation 1
2 Avertissements sur la puissance de l'informatique . . 2

Chapitre II : Historique et Justifications

1 Ma perception de l'informatique dans les hôpitaux . . 1
1.1 Le service informatique 1
a) Le personnel 1
b) Le matériel 1
c) Les logiciels 1
1.2 Les médecins de l'hôpital 1
1.3 Service informatique et les médecins 2
1.4 Le docteur Troch d'Ottignies 2
1.5 Conclusions 3

2 Historique du mémoire 3
2.1 Idées initiales 3
2.2 Choix du matériel 4
2.3 Choix de logiciel 4

3 Les options 5

Chapitre III : Les Concepts

1 Introduction 1

2 Hôpital, Services et Patients 1

3 Les Données 2
3.1 Champs, Mesures et Eléments-de-code 2
3.2 Dossiers et Pages-Structure 4

4 Illustration des concepts 5
4.1 Structure et Contenu. 5
4.2 Les différents champs possibles. 6
4.3 Utilisation du logiciel : 9
a) Soit un hôpital composé de services. 9
b) Exemples de structures de dossiers. 9
c) Exemples de structures de pages. 10

d) Scénario d'utilisation du logiciel. 12

5 Archivage et Journal 12

6 Sécurité 13

Chapitre IV : Le Modèle entité-association

1 Le modèle E-A : outil d'analyse conceptuelle 1
1.1 Introduction 1
1.2 Concept d'Entité 1
1.3 Concept d'Attribut 2
1.4 Concept de Contrainte d'intégrité 3
1.4.1 Notions. 3
1.4.2 C. I. sur des types d'entités 3
1.4.3 C. I. sur des types d'associations 3
1.4.4 C. I. sur des attributs. 4

2 Les graphes 4
Fig IV-1 5
Fig IV-2 5

3 Les entités 7

4 Les Associations et leurs Connectivités 11

Chapitre V: La logique du logiciel apparaissant dans le mode d'emploi

1 Introduction 1

2 Les fonctions 1
2.1 Définitions et remarques 1
2.2 Lecture du mot de passe 2
2.3 La table de travail 3
a) Description de la table 3
b) Les rubriques 4

I) Contenu 4
I-1) Le menu 4
I-2) Sélection d'une page 5
a) Sélection 5
b) Liste des patients du service visité . 6
c) Liste des services 7
d) Liste des pages du service visité . . . 7
e) Modification de la date et pagination . 8
I-3) Actions sur une page 8
a) Introduction 8
b) Création 9
c) Consultation, Modification 10
d) Suppression 10

II) Structure 11

II-1) Le menu 11
II-2) Pages 11
a) Introduction 11
b) Liste de pages 12
c) Les champs dans une page 13
d) Accès 14
II-3) Champs 14
a) Introduction 14
b) Champ : 'Nombre' 15
c) Champ : 'Suite de caractères' 16
d) Champ : 'Date' 16
e) Champ : 'Heure' 17
f) Liste des champs 18
II-4) Code 19
a) Introduction 19
b) Liste des codes 19
c) Modification d'un code 20

III) Impression 21
III-1) Le menu 21
III-2) Les Fonctions 22
a) Longueur d'une page 22
b) Liste des patients 22
c) Liste des services 23
d) Liste des pages 23
e) Liste des champs 23

IV) Divers 24
IV-1) Le menu 24
IV-2) Les Fonctions 24
a) Archivage 24
b) Reprise 25
c) Modification du mot de passe 25
d) Modification de la liste des patients . 26
e) Modification de la liste des services . 27

V) Sortie 27

2.4 Journal 27

Chapitre VI : Les fonctions et la base de données

1 Le Modèle des Accès Possibles (M.A.P.) 1
a) Introduction 1
b) Le schéma découlant du modèle E-A 4

2 Les fonctions et la BD 5
2.1 Définitions et remarques 5
2.2 Lecture du mot de passe 6
2.3 La table de travail et ses rubriques 8

I) Contenu 8
I-1) Le menu 8
I-2) Sélection d'une page 9
a) Sélection 9
b) Liste des patients du service visité . . . 10

c) Liste des services 12
d) Liste des pages du service visité 13
e) Modification de la date et pagination . . 15
I-3) Actions sur une page 17
a) Introduction 17
b) Consultation, Modification 17
c) Création 20
d) Suppression 20

II) Structure 20
II-1) Le menu 20
II-2) Pages 20
a) Introduction 20
b) Liste de pages 21
c) Les champs dans une page 22
d) Accès 24
II-3) Champs 25
a) Introduction 25
b) Champ : 'Nombre' 26
c) Champ : 'Suite de caractères' 26
d) Champ : 'Date' 26
e) Champ : 'Heure' 26
f) Liste des champs 27
II-4) Code 28
a) Introduction 28
b) Liste des codes 28
c) Modification d'un code 29

III) Impression 30
III-1) Le menu 30
III-2) Les fonctions 30
a) Longueur d'une page 30
b) Liste des patients 31
c) Liste des services 31
d) Liste des pages 31
e) Liste des champs 31

IV) Divers 31
IV-1) Le menu 31
IV-2) Les fonctions 31
a) Archivage 31
b) Reprise 31
c) Modification du mot de passe 32
d) Modification de la liste des patients . 33
e) Modification de la liste des services . 33

V) Sortie 33

2.4 Journal 33

3 Le Modèle d'Accès Généralisé (M.A.G.) : Le schéma . . . 34

CONCLUSIONS

1 Ce que m'a apporté le mémoire 1
2 Améliorations et extensions 1
3 Bilan du mémoire 1

Bibliographie 1

Introduction

1 L'idée du mémoire

On a observé deux grands agents qui font obstacle à une médecine de qualité ; c'est la communication et la précison des informations concernant les malades. Si, tout en respectant la vie privée des patients, nous pouvions développer un outil capable de contrecarrer ces facteurs de désordre, nous apporterions certainement un plus à la médecine actuelle.

L'idée de ce mémoire est l'étude d'un outil informatique permettant aux médecins d'un établissement hospitalier de conserver et de partager certaines connaissances relatives à un malade. Cette communication pourrait conduire à une meilleure adaptation des traitements au profit du malade.

L'outil devrait permettre un accès rapide à des données concises, précises et récentes. Chaque médecin pourrait traiter ses propres informations concernant ses patients, compulser les dossiers médicaux établis par d'autres médecins dans d'autres services.

Une première étude auprès des médecins, fit apparaître que chaque branche de la médecine a des besoins très spécifiques. Par conséquent, il n'est pas pertinent de créer un dossier médical standard. Il faudrait mieux fournir aux médecins la possibilité de créer un dossier médical sur mesure.

En conclusion, le logiciel à développer est un outil, facile à utiliser, de création de dossiers médicaux sur mesure.

2 Vocabulaire

Caractéristiques de l'information : Concise, précise ; ces mots définissent la précision et le caractère laconique de l'information, le logiciel en réalité ne peut le garantir, car ce point est laissé à l'appréciation du médecin. Le mot récent, fait référence à l'âge de l'information, cette dernière devra être disponible instantanément et au plus tard dans les 24 heures, sauf éléments extérieurs perturbateurs. Le terme 'accès rapide' désigne le temps qui sépare l'entrée d'une requête à un terminal et le moment où l'on voit apparaître la réponse à l'écran. Ce temps est lié à la machine utilisée.

Par les mots 'utilisation aisée', nous entendons que le logiciel doit être convivial. Comme il n'existe pas de normes en la matière, nous prendrons comme référence le logiciel **"FRAMEWORK"**. C'est un logiciel à services intégrés, il comprend un gestionnaire d'idées (traitement de texte avec table des matières), base de données (compatible DBASE-III), tableur...

Convivial : Un logiciel est convivial si son utilisation est aisée. Des commandes simples, des messages d'aide, des écrans clairs et peu fatigants... sont l'apanage d'un tel logiciel.

Par les mots 'adaptés aux besoins médicaux', nous entendons que le logiciel soit plus qu'un simple gestionnaire de fichiers. L'affichage des données doit être orienté vers une utilisation médicale.
Il doit tenir compte du problème de la confidentialité des informations. Ceci comprend le secret médical, qui impose l'emploi d'un mot de passe..., afin d'éviter que des personnes étrangères n'aient accès à des informations. Il faut permettre l'accès de parties de dossiers médicaux à des services non propriétaires des données.

Base de données (BD) : Une base de données est une collection d'informations de différents types, contenant des relations entre les types d'informations et ayant un caractère évolutif.
ex : Une base de données des animaux de la terre: il existe des insectes, des mammifères... Un type de relation serait 'vit sur la terre' qui rassemble tous les animaux vivant sur la terre.

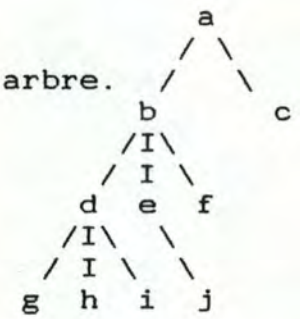
Gestionnaire de base de données : Nous entendons par là, un programme qui permet la création, la consultation, la mise-à-jour ou la suppression d'un dossier médical. Ceci est valable pour le contenu, i.e. les informations relatives à un malade, et pour la structure du dossier, i.e. la forme du dossier : libellés...

Identifiant : C'est une propriété d'un objet qui le caractérise de manière univoque.
ex : Le nom scientifique de chaque animal, un autre identifiant serait le code génétique de l'espèce.

Structure arborescente : Un arbre est composé d'une hiérarchie d'éléments appelés **noeuds**. Le plus grand niveau de cette hiérarchie possède un noeud unique appelé la **racine**. A l'exception de cette racine, chaque noeud est relié à un noeud d'un niveau supérieur, ce dernier est appelé son **père**. Aucun noeud ne peut avoir plus d'un père. Chaque noeud peut avoir des relations avec des noeuds de niveaux inférieurs, ces éléments sont appelés des **fils** de ce noeud. Il n'existe aucune relation entre des noeuds d'un même niveau. On appelle **feuille**, un noeud qui ne possède pas de fils. On dira que l'on remonte dans un arbre si l'on se rapproche de la racine.

Exemple de Structure Arborescente :

a : est la racine de l'arbre.
b,e,f : sont les noeuds de l'arbre.
c,d,g,h,i,j : sont les feuilles de notre arbre.
e est le père de g,h,i.
e est le fils de b.



3 Profil du mémoire

Chapitre I :

Etant donné le développement important de l'informatique actuelle, nous définirons les qualités que nous y attendons. Nous essayerons de rendre le lecteur plus sensible à l'informatique dans le domaine médical.

Chapitre II :

Nous développerons la chronologie des événements qui nous ont poussés à prendre des décisions importantes dans l'implémentation du logiciel. Nous verrons pourquoi nous avons été obligés de limiter l'implémentation du logiciel au développement de sa maquette.

Chapitre III :

Nous y définirons les concepts du problème et les besoins des utilisateurs.

Chapitre IV :

Pour avoir une vue plus formelle du problème, nous exprimerons sous la forme d'un modèle "entité-association" qui nous permettra, par la suite, de développer notre base de données.

Chapitre V :

Nous décrirons dans ce chapitre la structure logique située à la base du logiciel. Grâce à la description des écrans nous illustrerons et définirons les notions de rubrique et de fonction qui leurs sont associées.

Chapitre VI :

Attention, ce chapitre s'adresse plus spécifiquement au développeur du logiciel.
Utilisant le schéma "Entité-Association" et connaissant les fonctions à implémenter, nous décrirons complètement la base de données.

Nous finirons par une réflexion sur la réalisation du mémoire, ses extensions et les espoirs que nous y fondons.

Chapitre I : Considérations et Avertissements

1 De la nécessité d'une informatisation

Il est un fait que de plus en plus de médecins développent leurs propres applications informatiques pour la gestion de leurs dossiers...

Mais ces applications sont souvent trop spécifiques à leurs besoins et développées dans une optique à trop court terme, difficile à faire évoluer. De telles applications se transforment avec les idées des auteurs et sont, de ce fait, peu structurées. Pour ces raisons, la maintenance de tels programmes est complexe. On arrive à des situations où un utilisateur abandonnera un programme devenu trop compliqué pour en recommencer un nouveau et ne retrouvera que des données éparses.

Au sein de l'hôpital, nous voyons naître des petites entités qui ne peuvent communiquer entr'elles. Nous sommes devant un monde qui évolue de manière anarchique.

Il existe une demande dans le domaine de l'informatique médicale que les utilisateurs sont prêts à recevoir. Mais, comme on vient de le voir, leur laisser l'initiative est dangereuse. Il faut donc un organe centralisateur qui serait le logiciel que nous allons voir.

Mais si les médecins ne construisent plus leur logiciel, il faut les consulter et même les faire participer à la réalisation d'un tel logiciel. C'est pour cette raison que nous avons passé un bon nombre d'heures à les écouter.

Il résulte de cette étude préliminaire, qu'un tel outil doit permettre à un service de mieux soigner un patient.

Pour cela il doit conduire :

-à l'intégration de toutes les informations concernant un patient.

-au développement de traitements toujours mieux adaptés.

-à une remise en question quotidienne des méthodes utilisées.

-à l'obtention d'un rendement maximun des ressources dont une équipe dispose.

-à l'amélioration des connaissances de la discipline.

-à libérer en partie les médecins de la paperasserie.

- ...

L'outil à concevoir doit être simple d'emploi, rapide, efficace, souple et d'un coût acceptable. Il doit permettre au personnel de se sentir concerné, responsable et motivé par la réalisation d'un objectif fixé.

Une mise en commun des informations dégagées grâce à l'utilisation de l'outil, ne peut qu'améliorer le rendement et la qualité du travail de toute une équipe.

Un tel logiciel peut fournir en sous-produit un RCM ou des lettres types à envoyer à des confrères.

2 Avertissements sur la puissance de l'informatique

N'oublions jamais, que l'informatique doit être plus qu'un simple gadget moderne qui fait gagner du temps. En effet, elle doit permettre avant tout, au médecin et à son équipe d'atteindre un objectif, celui de **mieux soigner son patient**.

L'utilisation de tels outils nécessitera toujours un travail préliminaire afin de clarifier ses idées, mais aussi afin de se fixer des objectifs biens réfléchis. **Un travail non préparé ne peut fournir de bons résultats**.

Il n'y a pas d'"informaGique". Dans les dossiers développés par les médecins on ne trouve que ce qu'ils y ont mis! Ni plus, ni moins. Se fier à 100 % à l'ordinateur serait une gageure. Il faut que les utilisateurs prennent conscience que cacher ou détruire certaines informations peut avoir des conséquences fâcheuses. Les supports informatiques ne sont pas à l'abri de violations du secret.

Un tel logiciel ne doit pas faire disparaître le dialogue entre médecins, mais doit le promouvoir. Il y a ce côté informel du dialogue que l'on ne peut représenter en machine et qui est le propre de l'homme.

Le temps qu'un tel logiciel peut faire gagner à un utilisateur, doit lui permettre **d'être plus proche de son patient**.

Normalement, l'organisation des rapports au sein de l'hôpital ne devrait pas se voir changer, si ce n'est d'un point vue qualitatif.

Afin de faire un bon usage de ce logiciel nous ne pouvons qu'inviter l'utilisateur à lire les travaux du Docteur Francis ROGER. Ce dernier étant un spécialiste renommé dans le domaine des dossiers informatisés. Ces lectures permettront d'éviter les erreurs de jeunesse commises par un utilisateur non expérimenté.
Nous conseillerons la lecture de sa thèse [ROGER].

CHAPITRE II : Historique et Justifications

1 Ma perception de l'informatique dans les hôpitaux

1.1 Le service informatique

a) Le personnel

La fonction du service informatique de l'hôpital Saint-Joseph de Gilly est de faire tourner des programmes de gestion...

Ce service possède une position privilégiée au niveau des rapports entre les différents services et est bien accepté par ces derniers. Le dialogue entre utilisateurs et informaticiens est donc aisé.

Les informaticiens se sentent concernés et responsables des logiciels qu'ils développent. De plus le chef du service est proche du personnel, ce qui explique la bonne entente au sein du service.

b) Le matériel

Pour des raisons de suivis et de fiabilité, le service n'utilise que du matériel IBM. Deux types de machines sont employées : un IBM 38, avec une mémoire de masse de 2 Giga Bytes et des micro-ordinateurs de type PC d'IBM.

Le service informatique étant spécialisé dans des programmes de grosses gestions, il utilise essentiellement le 38.

c) Les logiciels

Le langage utilisé est le RPG III. Je garde de mauvais souvenirs de ce langage à la structure archaïque et étrange. Je désire souligner quand même que les outils de gestion de base de données sont très développés et d'une puissance peu commune.

Pour les PCs peu utilisés par le service informatique, les langages employés sont le BASIC et DBASE III.

1.2 Les médecins de l'hôpital

La réaction des médecins face à l'informatique est à classer en plusieurs catégories:

- L'informatique est étrange et pertubatrice.
- L'informatique peut tout résoudre.
- L'informatique, on la fait soi-même.
- L'informatique a des conséquences à long terme et doit se penser en groupe.

D'après ce que j'ai pu constater, les applications développées par des médecins sont bien adaptées à leurs besoins et sont la preuve d'une imagination très fertile. Mais au delà, il faut dire que ces logiciels sont mal conçus, difficiles à modifier, trop personnels et non documentés!

Maintenant, ces informaticiens-amateurs comprennent que l'informatique est plus qu'un gadget pratique, elle est un puissant moyen de traitement, de communication et d'innovation.

1.3 Service informatique et les médecins

Les médecins utilisateurs ne touchant pas à la programmation dialoguent aisément avec le personnel informatique afin de définir leurs besoins.

Les médecins qui touchent à l'informatique attendent souvent du service informatique des conseils pour leurs propres applications. Mais il y a toujours une rétention d'informations de la part des médecins qui veulent conserver leurs prérogatives et leurs libertés.

Mais, si le dialogue est aisé, le service informatique voit d'un oeil méfiant, le développement anarchique de logiciels sur micro. En effet, le jour où les informaticiens devront intégrer toutes ces petites entités, la chose ne sera pas aisée.

1.4 Le docteur Troch d'Ottignies

Ici, je voudrais faire part d'une expérience menée par le Docteur Troch qui est une réussite. Cette expérience ouvre une porte sur "l'informatique en blanc" au service du malade.

Ce médecin s'occupe d'un service d'hémodialyse occupant environ dix personnes. Ce dernier est habitué à la micro-informatique plus souple, moins onéreuse et donc plus abordable que la grosse informatique. Son expérience l'a conduit à des conclusions qu'il a mises en pratique.

"L'informatique doit être utilisée pour mieux soigner ses malades. C'est un instrument qui conduit à une remise en question permanente des pratiques médicales habituelles", nous dit-il.

Le service est géré et organisé comme une entreprise, afin d'obtenir un rendement maximum, par des méthodes équilibrées, intégrées, des tâches motivantes et responsabilisant les infirmières de son service.

Il désire ainsi mieux se consacrer à ses malades, en développant une informatique, qui à terme, s'adaptera mieux aux patients. Sa méthode devrait, selon lui, déboucher sur la médecine assistée par ordinateur.

1.5 Conclusions

Il existe un frein énorme à la "médecinetique": la volonté de faire de l'informatique sans informaticien. En effet, l'absence de méthodes de travail et d'une connaissance globale du monde informatique empêche l'optimisation des ressources informatiques. Souvent, chacun développe sa propre application, rendant impossible toute communication et toute évolution.

Ces médecins sont tiraillés entre leur liberté et la perspective d'un gain d'informations et d'efficacité dans le traitement des maladies, grâce à l'informatique.

En effet, la communication de renseignements généraux implique une sorte de standardisation, tels que les "Résumés Clinique Minimums" (sorte de synthèse d'un dossier médical) qui y seraient centralisés. Ces RCMs permettraient à l'Etat de surveiller les pratiques des médecins, qui perdraient dès lors un peu de libertés.

C'est pour cette raison entre autres, que les médecins veulent développer leur propre informatique. L'idée de 'club d'informatique en médecine' est bonne, mais sans un organe centralisateur servant de guide, l'expérience ne peut déboucher que sur un échec.

2 Historique du mémoire

2.1 Idées initiales

A l'origine, nous désirions étudier la réalisation d'un programme qui donnerait la possibilité aux médecins d'un établissement hospitalier de mettre en commun leurs connaissances relatives à un malade.

Cela reviendrait à fournir à chaque médecin un accès direct au dossier du malade. Les dossiers centralisés, permettraient de cette façon une meilleure distribution de l'information dans l'hôpital.

Pour la réalisation, nous désirions nous limiter à un système relatif à une seule branche. Nous avions choisi la réanimation pour le grand nombre de cas cliniques différents qui peuvent s'y présenter et en raison du faible nombre annuel de patients.

Afin de ne pas développer des hypothèses trop restrictives pour un système, il a fallu consulter différents médecins, représentatifs des différents dossiers médicaux possibles. Très vite il est apparu que chaque branche avait des besoins très spécifiques. De ce fait, il n'est pas possible de creér un dossier médical standard. Il faut donc fournir la possibilité d'en créer un sur mesure pour chacune des disciplines de l'hôpital, tout en lui permettant de se transformer au cours du temps.

En conclusion, le logiciel qui est étudié est un outil de création de dossiers médicaux sur mesure. C'est un programme de gestion de base de données où l'utilisateur peut afficher des données sous la forme qu'il préfère sans se préoccuper de l'accès aux fichiers.

Au départ nous pensions implémenter entièrement ce logiciel mais, suite à des ennuis rencontrés lors de l'utilisation d'outils logiciels, nous nous sommes contentés de concevoir le problème et d'implémenter les écrans dans une maquette. Il manque donc la programmation des procédures de gestion de la base de données.

2.2 Choix du matériel

Ce paragraphe s'adresse au lecteur initié à l'informatique.

Dans l'optique d'une centralisation des données, il est certain que le système 38 est mieux adapté que des micros. Mais un nombre de plus en plus élevé de médecins possèdent des PCs et désirent pouvoir travailler chez eux de la manière la plus indépendante possible du mainframe. La centralisation sur le 38 risquait de faire double emploi avec les logiciels et bases de données que développent les médecins de leur côté en réaction à une informatique trop lourde. La solution fut d'employer des cartes d'émulation qui permettent à un PC de se comporter comme un terminal du 38 et en même temps de garder ses caractéristiques propres. L'idée finale fut de créer des disques virtuels sur le 38, ces disques étant du format développé par les PC mais malheureusement inaccessibles à partir du 38. De plus l'accès à ces données est plus lent que l'accès à une disquette. La solution ne peut donc être retenue. Aucune solution n'a été trouvée actuellement.

L'idéal serait peut-être un ordinateur par service centralisant ses propres informations mais relié à d'autres par un réseau local, ce qui éviterait des copiages et des risques d'évolutions divergentes des bases de données centralisées et personnelles.

2.3 Choix de logiciel

Ce paragraphe s'adresse au lecteur initié à l'informatique.

Dans un premier temps, comme les idées n'étaient pas encore biens définies, nous pensions développer le logiciel sur le 38. Pour cela j'étudiais le RPG.

Ensuite quand l'idée se précisa et que nous décidâmes d'employer des PCs, le langage qui semblait le mieux adapté pour ses standards était le DBASE-III. C'est un outil dont l'usage est aisé, des instructions puissantes, permettant de développer rapidement une base de données. La mise au point est d'autant plus rapide que le langage est interprété. Mais le revers de la médaille est une lenteur d'exécution et la difficulté de créer des logiciels spécifiques nécessitant l'utilisation de routines de bas niveaux inaccessibles au DBASE-III. Pour remédier à cette lenteur, il existe des compilateurs de ce langage, mais ceci ne résout pas tous les problèmes soulevés.

Le "turbo Pascal" est un langage rapide, disposant d'instructions d'assez bas niveau pour faire tout ce qui était nécessaire. Mais il fallut très vite déchanter car s'il existe un gestionnaire de base de données pour le TURBO, il permet la gestion d'un nombre trop limité d'enregistrements.

Le choix se porta ensuite, sur le langage C. Ce langage, offre des instructions de niveaux encore plus bas que ceux du TURBO. De plus, le logiciel de base de données 'DBC' de Lattice-C est totalement compatible avec le système de DBASE-III. Ce qui est intéressant, en effet, DBASE-III est si bien développé que bien des logiciels intégrés sont compatibles avec sa structure de base de données. Pour fixer un choix, nous avons décidé de prendre FRAMEWORK comme référence standard de ces logiciels.

3 Les options

L'idée initiale était de créer un produit le plus standard possible. Les concepts du logiciel furent développés dans cette optique (cfr. chapitre III).

Nous désirions un logiciel facile d'utilisation, et rapide à apprendre. Pour être facile, nous avons regardé les logiciels qui existaient sur le marché. Nous vivons actuellemnt la mode du Macintosh avec sa souris et ses fenêtres. Nous avons délaissé la souris pratique à utiliser mais, encombrante sur un bureau. Par contre l'idée de fenêtre est excellente. Nous imiterons FRAMEWORK qui utilise des fenêtres sans faire appel à la souris. Nous avons donc repris les standards utilisés par ce logiciel. De plus ce logiciel est compatible avec DBASE III. A l'époque nous espérions encore pouvoir utiliser DBC de Lattice-C.

On peut remarquer que les options prises ici, n'interdisent pas l'utilisation de ce logiciel à des fins personnelles.

Dans un premier temps j'avais développé un prototype du logiciel en TURBO-PASCAL qui ne faisait que simuler les écrans. Le choix s'était porté sur le Turbo, car la programmmation en Pascal est plus rigoureuse qu'en C.

Sans rentrer dans la structure du programme-maquette, nous soulignerons que ce dernier a été conçu dans deux optiques. La première est de disposer d'outils permettant de modeler un logiciel à notre convenance. Ce qui explique que tous les écrans se trouvent dans un fichier séparé du programme principal qui d'une certaine manière est indépendant de ces écrans. L'autre optique est une implémentation aisée sur base de la maquette. En effet, les procédures sont prévues pour recevoir les programmes définitifs mais la malchance a voulu que je ne puisse pas les implémenter.

Une fois les écrans bien définis, je désirais revenir en C. Mais là je découvris des erreurs dans le compilateur. Si au début je pris le temps de les contourner, je dus avouer ma défaite vu le problème d'un 'Stack Overflow' récalcitrant.

Nous décidâmes alors de réduire l'application et d'utiliser le gestionnaire de base de données du Turbo. Mais là aussi il fallut abandonner tout espoir. En effet, toutes les tentatives effectuées pour intégrer les différentes parties du logiciel furent refusées par le système.

Nous dûmes nous résigner à une maquette écrite en Turbo Pascal.

Tous ces déboirs ne doivent pas faire penser que les micros sont des gadgets mals conçus. Le problème est que les langages que l'on peut trouver sont malheureusement bien souvent en retard sur les possibilités des micros et sur les nouvelles optiques de développement de logiciel. C'est une crise de jeunesse. Les micros restent pour moi l'outil par excellence pour développer le logiciel décrit dans ce mémoire.

A propos du chapitre VI ; Nous avions initialement décidé d'y ajouter les algorithmes d'accès à la base de données. Nous avions donc commencé à rédiger les différents algorithmes en ADL (cfr [HAINAUT]) mais une fois que je désirais les utiliser, je me rendis compte qu'ils étaient inutiles car trop loin des algorithmes définitifs. Donc, cette approche s'est révélée à la fois incomplète pour être utilisée telle quelle et à la fois trop lourde pour une lecture des spécifications. C'est pour ces raisons que nous avons décidé de simplifier le chapitre au niveau base de données.

Chapitre III : Les Concepts

1 Introduction

Dans ce chapitre nous allons introduire les concepts qui seront nécessaires à l'élaboration du logiciel.

2 Hôpital, Services et Patients

Nous avons un '**Hôpital**' ; c'est un organisme qui 'rassemble' une ensemble de 'Services'. Il est identifié par son nom.

Un '**Service**' est une unité de traitement d'informations pouvant : enregistrer, consulter, modifier, supprimer des données relatives à un patient ou être consulté à propos de ses données. Un service est identifié par son nom.

Un '**patient**' est un individu connu d'au moins un service de l'hôpital et à propos duquel un service peut enregistrer des informations. Un patient est identifié par son nom, son prénom et sa date de naissance.

Chaque service construit sa propre 'liste de patients' regroupant l'ensemble des patients qu'il '**connaît**'. Un service peut modifier sa liste :

- soit avec un patient déjà connu au sein de l'hôpital, i.e. appartenant à la liste des patients du service 'admission'.
- soit en créant un nouveau patient non encore connu du service 'admission'.

Un service peut encore supprimer un patient de sa liste, mais il ne peut modifier le nom, le prénom ou la date de naissance d'un de ses patients. Si un service désire modifier un de ces trois paramètres il devra passer par le service 'admission'. Cette option est prise afin d'éviter que tout le monde puisse modifier des informations nécessaires à tous.

C'est l'hôpital qui crée ou supprime un service. Il peut aussi changer le nom d'un service. En cas de suppression d'un service, toutes les données enregistrées par ce service sont perdues.

Il existe deux services particuliers que l'hôpital ne peut ni détruire ni modifier leur nom :

- Le service '**Admission**' : Sa liste des patients est la réunion de toutes les listes des différents services de l'hôpital. Cette liste est enrichie chaque fois qu'un service fait la connaissance d'un nouveau patient n'appartenant pas à cette liste. La destruction d'un patient dans la liste d'un service autre que celui d'admission n'a aucun effet sur la liste des patients de ce dernier.

Ce service est le seul à pouvoir modifier l'identifiant d'un patient ou faire disparaître définitivement un patient, il est nécessaire dans ce cas qu'aucun service ne fasse référence à ce patient.

Remarque : Si un seul utilisateur désirait utiliser le logiciel, il suffit qu'il s'identifie comme étant le service admission.

- Le service **'Commun'** : C'est un service mutant, du fait qu'il ne peut enregistrer des données, ni en modifier. Il sert de pot commun à tous les services, i.e. que chaque service peut y mettre une partie de l'information qu'il détient de manière à la rendre accessible à tous. Ce service ne peut donc être que consulté ou enrichi que par un service différent. Ce service commun permet à tout service qui le consulte, de disposer des informations relatives à un patient et considérées comme les plus pertinentes par les autres services. Ceci évite à un service chercheur d'informations d'aller glaner des données un peu partout.

L'hôpital et les services (à l'exception du service commun) sont les seules entités pouvant consulter ou modifier la base de données. Afin de préserver le secret médical, il faut fournir un 'mot de passe' pour accéder à chacun de ces derniers.

Lorsque l'hôpital crée un service, son mot de passe est mis à blanc. C'est au service de déterminer son mot de passe. Toutefois l'hôpital a la possibilité de mettre à blanc le mot de passe d'un service, cette possibilité est intéressante dans le cas où un service aurait oublié son mot de passe.

3 Les Données

3.1 **Champs, Mesures et Eléments-de-code**

Un service enregistre de l'information relative à un patient sous forme d'un dossier médical.

Données typées :

On suppose que toutes les données que l'on peut enregistrer à propos d'un patient appartiennent toujours à un type défini. Nous dirons que toutes ces données que nous appellerons **"Mesure"** sont d'un type dont la définition est appelée **"Champ"**.

Un **'Champ'** 'définit' une mesure selon un ensemble de caractéristiques :

- Un nom de champ.
- Un type de champ : - Numérique : Soit Entier.
  Soit Réel.
  - Suite de caractères.
  - Date.
  - Heure.
  - Code.

-le genre, qui peut-être dynamique ou statique, i.e. s'il faut ou non garder une trace des différentes mesures, en fonction du temps.

-Une borne inférieure et supérieure pour les types qui le permettent, i.e Numérique, Date, Heure.

-le caractère répétitif, i.e. si une mesure de ce type de champ peut prendre plusieurs valeurs à la fois.

-la valeur par défaut que doit prendre ce champ.

-Une ligne de commentaire apparaissant dans le bas de l'écran.

-Une unité, si nécessaire pour les champs numériques.

-Le caractère Commun : La définition d'un champ est mise à la disposition de tous dans le service 'Commun'. Ce qui est pratique dans le cas d'un code...

Remarques à propos de la destruction et de la modification d'un champ :

Une destruction de champ n'est effective que si aucune mesure ne se sert de ce champ. Dans le cas inverse, la destruction n'est que logique, i.e. que le champ est marqué, il n'apparaît plus dans la liste des champs courants, mais il existe encore pour permettre la relecture d'anciennes mesures.

L'utilisateur peut modifier un champ de deux manières :
- Soit une modification qui ne prend cours qu'à partir d'une certaine date.
- Soit une modification qui a des effets rétroactifs, i.e. qui modifie la définition même dans le passé.

En cas de modification, avec effets rétroactifs, de champs qui sont déjà employés, par des mesures, cette modification ne doit pas faire perdre de l'information.
Les modifications avec effets rétroactifs permises sont les suivantes:
-Modifier le nom d'un champ.
-Modifier le type entier en type réel, les autres passages d'un type à l'autre sont proscrits.
-Diminuer une borne inférieure.
-Augmenter une borne supérieure.
-Permettre le caractère répétitif à un champ qui ne l'avait pas.
-Modifier la ligne de commentaire.
-Modifier l'unité de mesure. (Attention aux effets rétroactifs)
-Définir un champ comme 'commun' s'il ne l'était pas. L'inverse étant prohibé.
-Modifier la valeur par défaut de ce champ.

----- °°°°° -----

Les Mesures :

Une '**Mesure**' est la plus petite quantité d'information définie par un champ qui peut 'concerner' un patient à une date donnée et 'enregistrée' par un service. Une mesure est identifiée par un patient, un service, une date et un champ qui la définit.

----- °°°°° -----

Les éléments-de -code :

Dans le cas où le type du champ est 'code', on lui associe un ensenble d'éléments appelés élément-code. Un élément-code est un libellé. Les éléments-code peuvent avoir une structure arborescente. Une telle organisation permet de représenter un code tel que le code ICD9CM. Une mesure relative à ce champ ne pourra prendre que des valeurs du code qui est un ensemble d'éléments-code.

A propos de l'évolution d'un code:
Un utilisateur peut détruire un élément du code. Les anciennes mesures faisant référence à cet élément sont toujours valables, la destruction réelle n'a lieu que s'il n'existe plus aucune mesure faisant référence à ce dernier, sinon la destruction n'est qu'apparente.
L'utilisateur peut modifier le libellé d'un code : Cette modification peut-être avec ou sans effets rétroactifs.

3.2 **Dossiers et Pages-Structure**

Un service regroupe des mesures dans une page, ce qui revient en fait à regrouper des champs. On appellera '**Page Structure**' cet ensemble de champs. C'est une manière de standardiser la présentation des mesures relatives à un patient.

Une 'Page Structure' est une grille de lecture des informations relatives à un patient, limitant la consultation, la modification à un nombre de mesures. Une page structure peut être composée de champs et de textes (en-tête de page...).

Cette page, ne correspondant pas nécessairement à un écran physique(écran du terminal), elle peut être plus grande ou même vide.

Une page utilisée, permet de consulter, de créer ou de modifier des mesures relatives à un patient à une date donnée.

Il existe deux types de pages :
- Le type 'Page normale' qui rassemble lors de son utilisation des mesures d'un patient enregistrées par un service à une date donnée.
- Le type 'Page Tableau' qui rassemble lors de son utilisation des mesures d'un patient enregistrées par un service sur un intervalle de temps donné.

Une page peut être dite '**Commune**', i.e. que le service qui en est l'auteur, donc le propriétaire décide qu'elle peut-être utilisée par tous les services sur un intervalle de temps défini et pour cela il la met dans le service commun. Tous les services peuvent désormais s'en servir pour consulter la base de données du service qui en est le propriétaire sur l'intervalle de temps permis.

Un service peut aussi décider qu'une page n'est accessible que par un service en particulier. Il détermine un intervalle de temps sur lequel le service étranger peut consulter cette page et un autre intervalle sur lequel le service étranger peut modifier sa base de données par le biais de cette page.

Pour ce qui est de la destruction d'une page ; Une page peut soit être détruite définitivement ou soit être 'marquée'. Dans ce dernier cas la page bien que n'apparaissant plus dans la liste habituelle des pages, existe toujours et peut encore être utilisée, ce qui permet, par exemple ,de faire disparaître une page ayant une présentation désuète, mais pratique pour la lecture d'anciennes données.

On appelle 'page récapitulative', une page qui reprend tous les champs définis par le service propriétaire à une date donnée. Cette page est crée dès qu'un service existe, sa structure ne sera modifiée que si l'on ajoute ou détruit un champ dans le service et ne sera détruite que si l'on détruit le service. Cette page ne peut être mise dans le service commun. A chaque date de visite correspond un contenu différent.

Le dossier médical d'un patient pour un service donné est l'ensemble des mesures enregistrées par le service concernant ce patient. Ces mesures pouvant être consultées par le biais de pages-structure : Elles sont des fenêtres sur la base de données, elles permettent de consulter un sous-ensemble des mesures relatives à un patient à une date donnée.

Donc pour un service donné, la structure est indépendante du patient tandis que le contenu est propre à chaque patient.

4 Illustration des concepts

4.1 Structure et Contenu.

Illustrons la distinction entre Structure et contenu.

a) La structure :

La structure du dossier est en quelque sorte un formulaire, une grille de lecture.

Ce sont les champs qui sont soulignés.
On peut ajouter aux champs du texte c'est le cas de l'entête : 'Dossier médical'.

b) Le contenu :

```
                    Dossier medical :

Service : Anesthésie
          ----------

Nom : Hivalan
      -------

Prénom : Paul
         ----

Chirurgien : Gerard Manvussa
             ---------------

Médecin traitant : Axel Errateur
                   -------------
```

Le contenu du dossier est constitué des noms soulignés.
Ce sont les mesures qui sont soulignées.

4.2 Les différents champs possibles.

Illustration des paramètres d'un champ:

-le type de la donnée (numérique {entier,réel}, suite de caractères, date, heure ou code).

Exemples :
* Numérique : Le poids d'un malade : 85,6 : valeur réelle.
Le numéro du patient : 123456 : valeur entière.

* Suite de caractères :
'Ne fume pas et a un très bon moral'.

* Date : 6/6/1944

* Heure : 6:00:00

* Code : Exemple de valeurs d'un code décrivant le réveil d'un patient.
2 Agité
3 Rapide

----- ----- -----

-le genre, qui peut-être **dynamique** ou **statique**, i.e. s'il faut ou non garder une trace des différentes mesures, en fonction du temps.

Exemples :

* Statique : La date de naissance : Si elle change, c'est qu'elle était érronée et son ancienne valeur ne nous intéresse pas.

* Dynamique : Le poids du patient. Le médecin est intéressé par l'évolution du poids de son client.

----- ----- -----

-Une borne inférieure et supérieure pour les types qui le permettent.

Exemple :
* Numérique : Le poids d'un malade
Un poids doit être compris entre 0 et 400.

* Date : Une date minimum et une date maximum.

* Heure : Une heure minimun et une heure maximun.

----- ----- -----

-le caractère répétitif, i.e. si une mesure de ce type de champ peut prendre plusieurs valeurs à la fois.

Exemple de champ répétitif :

Traitements subis:- Ablation de l'estomac.
- Ablation de l'appendice.

----- ----- -----

-la valeur par défaut que doit prendre ce champ.

Exemple :
* Poids : 0 ; indique que le champ n'a pas été rempli.(convention de l'utilisateur).

----- ----- -----

-Une ligne de commentaire apparaissant dans le bas de l'écran.

Exemple :

* 'Un taux d'albumine › x est dangereux'

----- ----- -----

- Une unité, si nécessaire.

Exemple :

* Poids : 'Kg'

----- ----- -----

- le caractère commun, c'est-à-dire, si le champ est accessible par le service commun.

Exemple :

* Un champ de type code : 'Les interventions chirurgicales et obstétricales'.

----- ----- -----

-Si le type du champ est 'code' : Le nom du code auquel il fait référence.

Chaque utilisateur peut définir son propre code et le mettre à la disposition de tous en lui donnant la désignation 'commun'. Chaque utilisateur pourra l'employer, mais aucun ne pourra le modifier, à l'exception de son propriétaire. Ce sera par exemple le cas pour le code ICD-9-CM ou pour le code INAMI qui sera mis à la disposition de tous par le service admission par exemple. De cette manière tout le monde utilise le même code et est à l'abri de modifications intempestives, car seul le propriétaire peut le modifier, en l'occurrence le service admission. Le service utilisateur n'aura plus qu'à utiliser ce champ de type code.

Exemple :
* Code : type de réveil

1 Calme.
2 Agité.
3 Rapide.
4 Lent.

* Le code ICD 9 CM.
extrait :
...
754.8 Anomalie musculo-squelettique non tératogène, congénitale,autre.
754.81 Thorax en entonnoir congénital.
754.82 Thorax en carène, congénital.
754.89 Anomalie musculo-squelettique non tératogène, congénitale, autre.
754.891 Anomalie de la paroi thoracique, congénitale.
754.892 Arthrogrypose multiple, congénitale.
...

----- ----- -----

Exemple de modification du code 'type de réveil' :

Destruction:
Le service décide que le libellé 'lent' doit disparaître, les mesures du type code 'type de réveil' ayant cette valeur garde cette valeur bien qu'elle n'existe plus.

Modification:
RETROACTIF : Le service se rend compte que le libellé 'Calme' avait été orthographié 'clme', il décide d'effectuer une modification avec effets rétroactifs, désormais les mesures qui avaient pour valeur 'clme' apparaîtront avec la valeur 'calme'.

NON RETROACTIF : Le service décide de remplacer, de manière non rétroactive le libellé 'agité' par 'très agité' et d'ajouter un libellé 'faiblement agité'. Les mesures qui avaient pour valeur 'agité' gardent cette valeur.

4.3 Utilisation du logiciel :

a) Soit un hôpital composé de services.

- Le service commun : Il dispose de toutes les informations d'ordre général concernant les patients.

- Admission : Il enregistre les données générales sur le patient. Nous y trouverons les données administratives. Il est le seul à pouvoir modifier ou supprimer un patient.

- Les services spécialisés :

  - Anesthésie.
  - Cardiologie.
  - Othopédie.
  - ...
  - laboratoire.

b) Exemples de structures de dossiers.

Les dossiers se décomposent en pages comme suit :

a) Le service commun :
---
1 Les caractéristiques du patient :
2 Le résumé des précédentes visites :
 2.1 Admission :
 2.2 Sortie :
 2.3 Diagnostics et complications :

b) Le service admission :
---
1 Caractéristiques du patient.
2 Entrée:
3 Déplacement:
4 Sortie:

c) Le service anesthésie:
---
1 Antécédents:
2 Caractéristiques.
3 Descriptions:
 3.1 Préopératoire.
 3.2 Intervention.
 3.3 Réveil.
4 Remarque.

d) Le laboratoire :
---
1 Sang.
2 Urine.

c) Exemples de structures de pages.

a) Service commun :

1 Les caractéristiques du patient:

Caractéristiques :
Nom :
Prénom :
Adresse :
N° :
Ville :
Code Postal :
Numéro National :
...

2 Le résumé des précédentes visites :

Admission :
Date d'admission :
Heure d'admission :
Type d'admission :
Médecin traitant :
...

b) Admission:

1 Les caractéristiques du patient:

Caractéristiques :
Nom :
Prénom :
Adresse :
Rue :
N° :
Ville :
Code Postal :
Numéro national :
Mutuelle :
Profession :
...

2 Entrée :

Admission :
Date d'admission :
Heure d'admission :
Type d'admission :
Médecin traitant :
...

...

Nous voyons que les caractéristiques du patient pour l'admission, sont plus fouillées que celles du service commun.

d) Scénario d'utilisation du logiciel.

1) Un patient arrive malade à l'hôpital :
A l'admission on remplit le dossier admission, indiquant s'il n'est pas encore connu, son nom etc.... Les données enregistrées sont en partie disponibles dans le dossier du service commun.

2) Supposons que notre malade souffre d'une appendicite. Il doit donc subir une intervention chirurgicale. Le chirurgien concerné feuillette les résumés des visites de son patient dans le service commun. Aucun antécédent négatif.

3) Chirurgien et anesthésiste se concertent pour décider de l'opération. Ce dernier fait analyser le sang du patient par le laboratoire. Pour ce faire, il définit une page de son dossier médical comme accessible au laboratoire.

4) Le laboratoire remplit cette page avec les mesures concernant l'échantillon.

5) Notre anesthésiste découvre que le malade est un fumeur de grande classe, il devra donc prendre ses dispositions.

6) Fort de ces nouvelles connaissances, nos médecins soignent le patient.

7) Après ce séjour, le chirurgien remplit le dossier du malade, dont une partie des informations sera accessible à tous grâce au service commun.

5 Archivage et Journal

Les informations enregistrées par un service pendant un an peuvent être archivées. L'archivage consiste en un stockage de toutes les informations nécessaires pour reconstituer la base de données sur cette période, i.e. la liste des patients que le service a connu, les mesures qui concernaient ces patients, les champs qui définissaient ces mesures et enfin les pages qui sont utilisées.

En plus de cette possibilité d'archivage, il existe la notion de journal. Ce dernier contient toutes les modifications effectuées sur la base de données.

Il a plusieurs fonctions :

- La première est de permettre de reconstituer la base de données en cas de pertes partielles ou totales de cette dernière. Nous ne détaillerons pas ce point, car il est lié au gestionnaire de base de données utilisé.

- La deuxième fonction est celle de preuve, il permet par exemple, de vérifier qu'une donnée à été modifiée au cours d'une journée. Tout ceci dépend de la durée sur laquelle s'étend le journal, de quelques jours à plusieurs semaines... Ce sont des décisions à prendre lors de l'implémentation finale.

Du fait de l'importance de ce journal, il est certain que son accès doit être limité au service **Hôpital**.

6 Sécurité

La sécurité de la base de données, est orientée vers deux types de personnes; primo les étrangers à toutes activités médicales et secondo les personnes appartenant au corps médical mais ne faisant pas partie de la section ayant développé la base de données.

Dans le premier cas, la sécurité est préservée par l'emploi d'un mot de passe. Chaque service ayant développé sa base de données en possède un seul.
En plus de ces mots de passe, les fichiers seront codés dans la base de données.

Dans l'autre cas, nous avons voulu permettre l'échange d'informations entre des services différents, mais tout en laissant au propriétaire de l'information la possibilité de ne délivrer qu'une partie des informations dont il dispose. Pour ce faire à chaque noeud-écran est associé une liste des services pouvant accéder à ce dernier. Un liste par défaut est établie pour tous les écrans d'un type, mais il est possible de raffiner le choix des personnes pouvant accéder à un dossier particulier. Ces accès peuvent être des consultations ou des modifications d'un écran appartenant à un dossier. Ce qui est pratique pour l'échange d'informations entre le laboratoire et un service lui ayant demandé une analyse, en évitant des recopies fastidieuses et sources d'erreurs.

Chapitre IV : Le Modèle entité-association

1 Le modèle E-A : outil d'analyse conceptuelle

1.1 Introduction

Désirant une approche plus rigoureuse nous avons décidé de formaliser les concepts précédemment définis.

Nous avons choisi l'approche Entité-Association (E-A) pour l'audience croissante qu'elle rencontre parmi les spécialistes des systèmes d'informations et des bases de données.

Nous allons définir quelques concepts clefs, mais ceci est trop succinct et nous ne pouvons qu'inviter le lecteur à consulter [BODART-PIGNEUR].

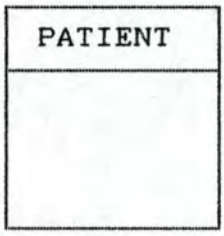
1.2 Concept d'Entité

Une ENTITE est une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations. Une entité n'existe en tant que telle que par rapport à un individu ou un groupe qui la considère comme un tout, lui confère une existence autonome et la distingue d'autres entités et de son environnement. Une entité peut posséder des attributs.

Un type d'entité : classe de toutes les entités possibles du réel perçu qui vérifie la définition constitutive du type. Les types d'entité ne constituent pas nécessairement des classes disjointes : une même personne pourraît être employée, actionnaire et cliente de la même firme.

Une occurrence d'un type d'entité : est une entité (individuelle) d'un type donné, i.e. un élément de la classe constituée par ce type. Les expressions "entité" et "occurrence d'un type d'entité" sont donc équivalentes.

Notation et représentation graphique : On représente une entité par un rectangle comportant un cartouche où figure le nom du type d'entité.



1.3 Concept d'Attribut

Un ATTRIBUT est la caractéristique ou qualité d'une entité ou d'une association. Elle peut prendre une ou plusieurs valeurs ou groupe de valeurs.

Propriétés d'un attribut :

* Attribut simple ou répétitif:

Un attribut est simple si pour une occurrence d'un type d'entité ou d'association, il ne peut prendre qu'une seule valeur.

Il est répétitif si pour une occurrence d'un type d'entité ou d'association, il peut prendre plusieurs valeurs pour un même type.

* Attribut simple ou décomposable :

Un attribut est décomposable si à une occurrence d'un type d'entité ou d'association, il fait correspondre un groupe de valeurs de types différents et peut être décomposé, au plus, en autant d'attributs qu'il y a de types différents dans le groupe de valeurs.

Un attribut non décomposable est élémentaire.

* Attribut obligatoire ou facultatif :

Un attribut peut-être obligatoire si chaque occurrence de l'entité ou association à laquelle il appartient implique qu'il ait nécessairement une valeur.

Il est facultatif s'il n'est pas nécessaire qu'il prenne une valeur.

Représentation graphique :

1.4 Concept de Contrainte d'intégrité

1.4.1 Notions.

Définition : Une contrainte d'intégrité (C.I.) est une propriété, non représentée par les concepts de base du modèle, que doivent satisfaire les informations appartenant à la mémoire du système d'information.

On distingue les contraintes **statiques** et les contraintes **dynamiques**.

Contrainte d'Intégrité Statique : Propriété que doit être vérifiée à tout moment : i.e. indépendamment des changements d'état de la base de données.
ex: Date de naissance ‹ Date de mariage.

Contrainte d'Intégrité Dynamique : Propriété qui définit la validité des changements d'état de la base de données. C'est une règle de transition qui définit les séquences possibles des changements d'état de la base de données.
ex : Une personne ne peut devenir divorcée que si elle était mariée précédemment.

1.4.2 C. I. sur des types d'entités

Contrainte d'Existence : En plus de la définition d'un type d'entité qui caractérise la condition d'existence d'une occurrence de ce type, il peut y avoir des contraintes qui lient la validité de l'existence d'une entité à d'autres éléments de la structure de données.
ex : Une voiture ne peut exister sans appartenir à quelqu'un.

Identifiant d'un type d'entité : C'est un groupe d'un ou plusieurs attributs, tel qu'à chaque combinaison de valeurs prises par ce groupe correspond au plus une entité de ce type. Un T.E. peut-être doté de plus d'un identifiant.

Notation : Lorsqu'il n'y a pas d'ambiguïté, on soulignera les attributs identifiants d'un T.E.

1.4.3 C. I. sur des types d'associations

Contrainte d'Existence : Elle exprime le fait que l'existence d'une association dépend d'éléments autres que ceux spécifiés dans la définition du T.A..

Contrainte d'exclusion : La participation d'une occurrence d'un T.E. dans une occurrence d'un T.A. exclut sa participation dans une occcurrence d'un autre T.A.

ex : Une occurrence de type 'Animal' est soit associée à une autre occurrence de type 'Mammifère' par une association de type 'An-Ma' ou de type 'Poisson' par une association de type 'An-Po'...

Contrainte d'Inclusion : La participation d'une occurrence d'un T.E. dans une association inclut sa participation dans une occurrence d'une association d'un autre type.

Identifiant d'une association : Par définition, une association d'un type donné sera identifiée par les identifiants des entités sur lesquelles elle est définie. En effet, l'existence d'une association est contingente à l'existence des entités qu'elle met en correspondance.

1.4.4 C. I. sur des attributs.

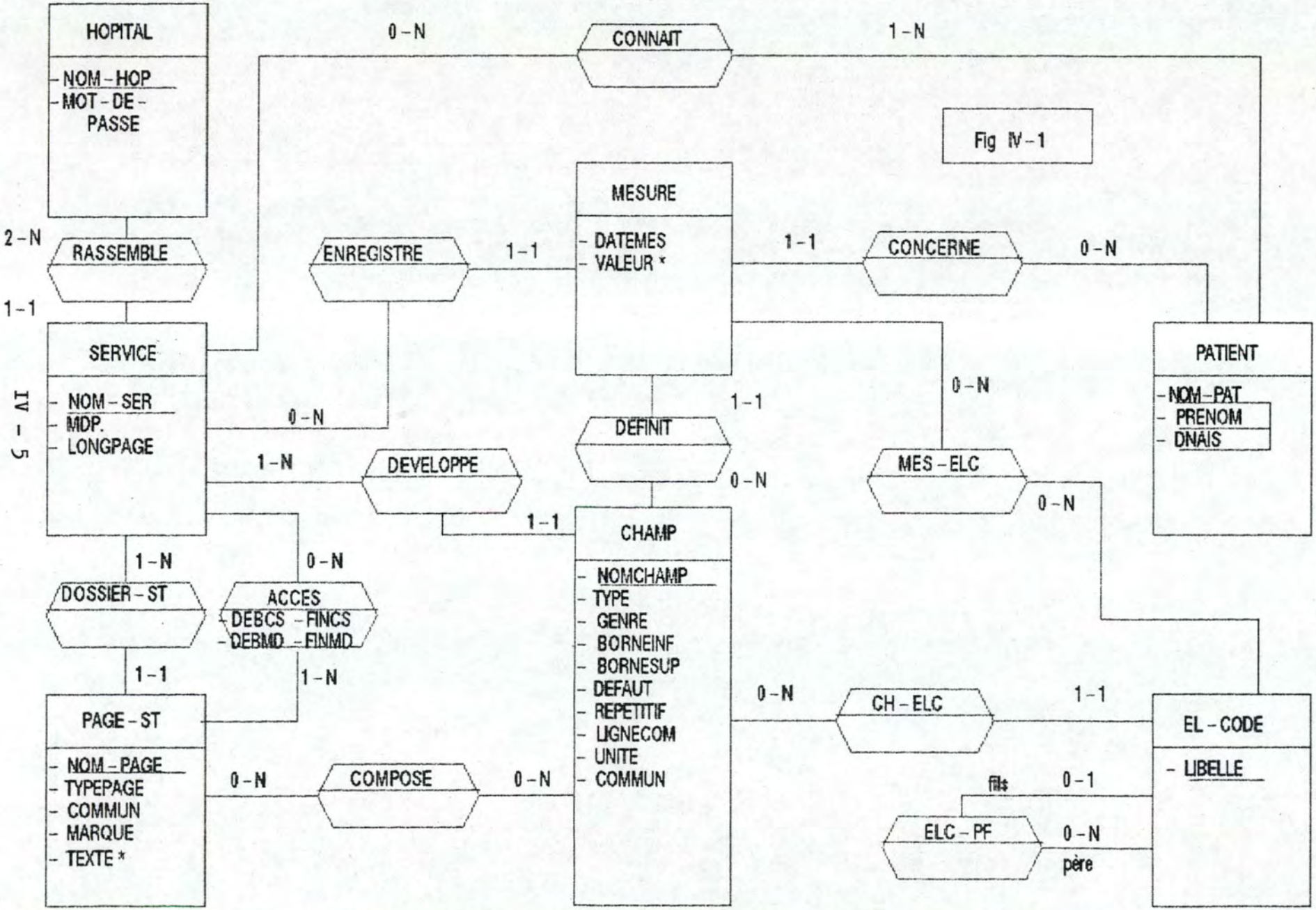
Contrainte de valeur : Elle définit
- Soit l'ensemble des valeurs que peut prendre un attribut.
- Soit les valeurs qu'un attribut prend en fonction des valeurs prises par d'autres attributs.

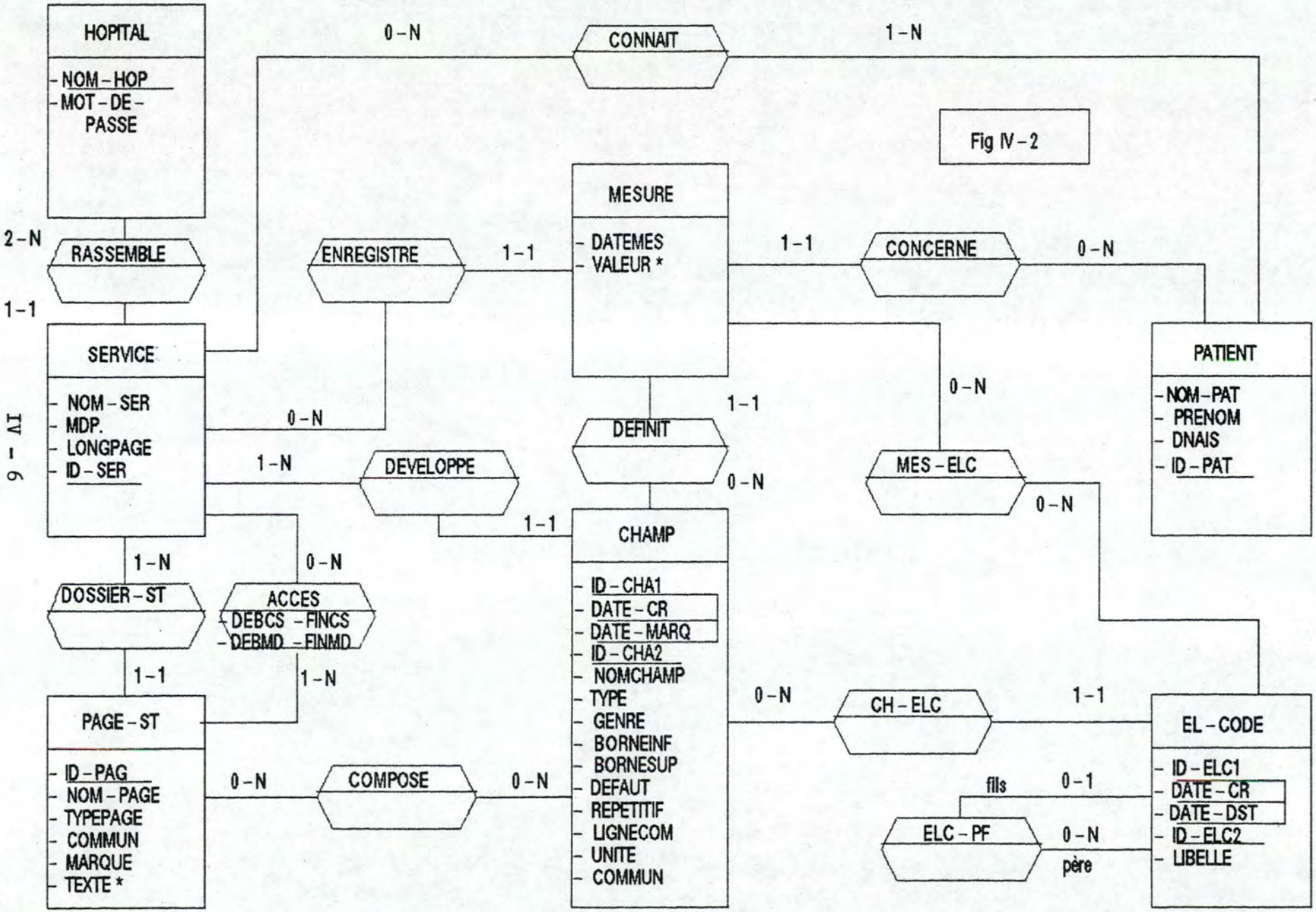
2 Les graphes

Nous allons trouver à la figure IV-1 une modélisation des concepts développés au chapitre précédent.

Mais certains concepts propres à l'implémentation se doivent d'y être représentés, c'est la figure IV-2. Nous trouvons des identifiants cachés à l'utilisateur permettant par exemple de modifier le nom d'un champ sans perdre la référence à ce champ.

Les entités et associations définies par la suite feront référence à cette figure.


HOPITAL
- NOM-HOP
- MOT-DE-PASSE
0-N
CONNAIT
1-N
Fig IV-1
MESURE
- DATEMES
- VALEUR *
2-N
RASSEMBLE
ENREGISTRE
1-1
1-1
CONCERNE
0-N
1-1
SERVICE
- NOM-SER
- MDP.
- LONGPAGE
0-N
1-1
DEFINIT
0-N
PATIENT
- NOM-PAT
- PRENOM
- DNAIS
1-N
DEVELOPPE
1-1
CHAMP
- NOMCHAMP
- TYPE
- GENRE
- BORNEINF
- BORNESUP
- DEFAUT
- REPETITIF
- LIGNECOM
- UNITE
- COMMUN
MES-ELC
0-N
1-N
DOSSIER-ST
0-N
ACCES
- DEBCS - FINCS
- DEBMD - FINMD
1-1
1-N
PAGE-ST
- NOM-PAGE
- TYPEPAGE
- COMMUN
- MARQUE
- TEXTE *
0-N
COMPOSE
0-N
0-N
CH-ELC
1-1
EL-CODE
- LIBELLE
fils
0-1
ELC-PF
0-N
père


HOPITAL
- NOM - HOP
- MOT - DE - PASSE
0 - N
CONNAIT
1 - N
Fig IV - 2
MESURE
- DATEMES
- VALEUR *
2 - N
RASSEMBLE
ENREGISTRE
1 - 1
1 - 1
CONCERNE
0 - N
1 - 1
SERVICE
- NOM - SER
- MDP.
- LONGPAGE
- ID - SER
0 - N
1 - 1
DEFINIT
0 - N
PATIENT
- NOM - PAT
- PRENOM
- DNAIS
- ID - PAT
1 - N
DEVELOPPE
0 - N
MES - ELC
0 - N
1 - 1
CHAMP
- ID - CHA1
- DATE - CR
- DATE - MARQ
- ID - CHA2
- NOMCHAMP
- TYPE
- GENRE
- BORNEINF
- BORNESUP
- DEFAUT
- REPETITIF
- LIGNECOM
- UNITE
- COMMUN
1 - N
0 - N
DOSSIER - ST
ACCES
- DEBCS - FINCS
- DEBMD - FINMD
1 - 1
1 - N
PAGE - ST
- ID - PAG
- NOM - PAGE
- TYPEPAGE
- COMMUN
- MARQUE
- TEXTE *
0 - N
COMPOSE
0 - N
0 - N
CH - ELC
1 - 1
EL - CODE
- ID - ELC1
- DATE - CR
- DATE - DST
- ID - ELC2
- LIBELLE
fils
0 - 1
ELC - PF
0 - N
père
IV - 9

3 Les entités

ENTITE : CHAMP

Définition : C'est une définition d'un "type" de données, (**définition-de-mesure**) indiquant que la mesure y faisant référence est un entier, réel, suite de caractères...

Identifiant(s) : Il en exitse deux :

1)ID-CHA1 : Identifiant interne associé au champ, il permet à l'utilisateur de modifier le nom du champ et garder une référence constante à ce champ.

+ **DATE-CR** : Date à partir de laquelle le champ peut-être utilisé.

+ **DATE-MQ** : Date à laquelle le propriétaire de ce champ a marqué ce champ.

+ l'identifiant d'un SERVICE.

2)ID-CHA2 : Identifiant interne de chaque version du champ.

+ l'identifiant d'un SERVICE.

----- °°°°° -----

Attributs : **NOMCHAMP** : Le nom du champ.

TYPE : Décrit le type du champ, les différents types reconnus sont: entier, réel, chaîne de caractères, date, heure et code.

GENRE : Statique ou dynamique.

-statique : On ne garde pas une trace des différents contenus de la variable.

-dynamique : On garde une trace des contenus de la variable en fonction du temps.

BORNEINF : Ne sert que pour les types différents de 'suite de caractères'. C'est une borne inférieure, en-dessous de laquelle la valeur d'une mesure y faisant référence ne peut descendre.

BORNESUP : C'est une borne supérieure, que le contenu d'une mesure y faisant référence ne peut dépasser. Pour une chaîne de caractères c'est le nombre maximum de caractères qu'elle peut contenir.

DEFAUT : Contient une valeur par défaut, c'est la valeur que contient une mesure si elle n'a pas été remplie.

REPETITIF : Indique si le champ peut prendre plusieurs valeurs simultanément.

LIGNECOM : Ligne de commentaire associée au champ.

UNITE : Une suite de caractères associée aux champs numériques définissant une unité.

COMMUN : Indique que la définition se trouve dans le service commun. Toute modification d'un champ commun se répercute dans le service commun.

----- °°°°° -----

Contraintes d'Intégrité :

DATE-CR & DATE-DST : La date de Création doit toujours être strictement plus grande que la date de destruction. Car on ne mémorise pas des champs construits et détruits au cours de la même journée. Si un champ est détruit puis reconstruit le jour même, on considère que nous avons une nouvelle version.

NOMCHAMP : Il ne peut exister deux champs au sein d'un même service qui possèdent un nom de champ identique pour des ID-INT1 différents.

BORNEINF & BORNESUP : La valeur de BORNEINF doit toujours être plus petite ou égale à la borne supérieure.

DEFAUT : Le contenu de ce champ doit toujours être compris BORNEINF et BORNESUP.

Il existe toujours un champ "visite" de type 'temps' associant une heure et une date.
Le type 'temps' intègre à la fois la notion d'heure et de date, elle n'a qu'une utilité interne.

---------*****----------

ENTITE : EL-CODE (élément-code)

Définition : C'est un libelle appartenant à un code défini par un champ de type code.

Identifiant(s) : Il en existe deux :
1 ID-ELC1 : Identifiant interne associé à l'élément du code, il permet à l'utilisateur de modifier le nom du libellé et garder une référence constante à ce champ.

+ DATE-CREATION : Date à partir de laquelle l'élément peut-être ultilisé.

+ DATE-DESTRUCTION : C'est la date à laquelle le service a détruit cet élément.

+ l'identifiant d'un SERVICE.

+ Un nom de champ de type 'CODE'.

2 ID-ELC2 : Identifiant interne de chaque version de l'élément du code.

+ l'identifiant d'un SERVICE.

+ Un nom de champ de type 'CODE'.

Attributs : LIBELLE : Libellé de l'élément du code.

----- °°°°° -----

Contrainte d'intégrité.

DATE-CREATION & DATE-DESTRUCTION : La date de Création doit toujours être strictement plus grande que la date de destruction. Car on ne mémorise pas des EL-CODE construits et détruits au cours de la même journée.

Conseil : Il est déconseillé à l'utilisateur de donner à un EL-CODE des libellés identiques car cela peut entraîner une perte d'informations, ces deux EL-CODE évoluant chacun de leur côté.

---------*****----------

ENTITE : HOPITAL

Définition : Organisme rassemblant un ensemble de services.

Identifiant(s) : **NOM-HOP** : Le nom de l'hôpital.

Attributs : **MOT-DE-PASSE** : Le mot de passe pour accéder aux différentes fonctions réalisables par l'hôpital sur la base de données.

---------*****----------

ENTITE : MESURE

Définition : La plus petite information pouvant être contenue dans un dossier médical.

Identifiant(s) : **DATEMES** : Date à laquelle a été prise la mesure.

+ L'identifiant d'un SERVICE.
+ L'identifiant d'un PATIENT.
+ l'identifiant d'un CHAMP.

Attributs : **Valeur** : C'est une valeur qui a été enregistrée lors d'un examen d'un patient, à une date donnée et doit être lue selon un CHAMP qui la définit.

---------*****----------

ENTITE : PAGE-ST

Définition : Elément constitutif d'un dossier médical, regroupe un ensemble de champs. Il peut contenir aussi des suites de caractères.

Identifiants :

* **ID-PAG** : Identifiant interne à la base de données, permet à l'utilisateur de changer le nom du champ sans devoir en créer un nouveau.

+ L'identifiant d'un SERVICE.

Attributs :

* **NOMPAGE** : Le nom de la page.

* **TYPEPAGE** : Un noeud peut contenir des données relatives à une date. Cest le type "normal".
Il peut aussi contenir des données relatives à un intervalle de temps. C'est le type "tableau".

* **COMMUN** : Une page peut ou non apparaître dans le service commun.

* **TEXTE** : C'est un attribut contenant l'ensemble des lignes de texte contenues dans une page.

* **MARQUE** : Une page peut-être marquée, ce qui indique qu'elle a été détruite non physiquement mais logiquement, i.e. il est encore possible de la rappeler pour l'utiliser.

* **TEXTE** : Item répétitif contenant la partie texte d'une page.

---------*****----------

ENTITE : PATIENT

Définition : Toute personne au sujet de laquelle un service désire enregistrer de l'information.

Identifiant(s) : **ID-PAT** : Identifiant interne à la base de données d'un patient, ce qui permet de modifier le nom, le prénom ou la date de naissance d'un patient sans perdre sa référence.

Attributs :
* **NOM** : Le nom du patient.
* **PRENOM** : Le prénom du patient.
* **DNAIS** : La date de naissance du patient.

---------*****----------

ENTITE : SERVICE

Définition : Unité de travail ayant la possibilité de créer, de modifier ou de consulter ou d'être consultée à propos de dossiers médicaux relatifs à des patients de l'hôpital.

Identifiant(s) : **ID-SER** : Identifiant interne à la base de données permettant d'identifier un service indépendant de son nom, de cette manière la modification du nom d'un service n'entraîne pas une perte de référence.

Attributs :
* **NOM-SERV** : Le nom du service.

* **MDP** : Un mot de passe.

* **LONGPAGE** : longueur d'une page de papier sur imprimante.

---------*****----------

4 Les Associations et leurs Connectivités

ASSOCIATION : ACCES

Définition : Associe un service à une page dont il n'est pas l'auteur-propriétaire, dans une relation définissant le type d'accès permis.

Attributs :

* **DEBCS** : Date à partir de laquelle le service associé à la page peut consulter cette dernière.

* **FINCS** : Date à partir de laquelle le service ne peut plus consulter la page.

* **DEBMD** : Date à partir de laquelle le service associé à la page peut modifier cette dernière.

* **FINMD** : Date à partir de laquelle le service ne peut plus modifier la page.

Ceci concerne la consultation d'une partie du dossier médical d'un patient, par le biais d'une page.
Remarque : les bornes sont comprises dans l'intervalle.

Connectivité :

- SERVICE : 0-N ; Un service existe indépendamment du fait qu'il peut consulter une page d'un autre service. Mais il peut avoir accès à plus d'une page.
- page : 1-N ; une page est associée à tous les services par l'association ACCES.

---------*****----------

ASSOCIATION : COMPOSE

Définition : Associe un champ à une page.

Attributs :

Connectivité : PAGE : 0-N : Une page peut exister sans faire référence à un champ et peut faire référence à plusieurs.
CHAMP : 0-N : Un champ peut exister indépendamment d'une page et peut intervenir dans plusieurs pages.

---------*****----------

ASSOCIATION : CONCERNE

Définition : Associe une mesure à un patient.

Attributs :

Connectivité : PATIENT : 0-N : Un patient peut exister sans qu'on ait enregistré des mesures à son sujet. Mais souvent on enregistrera plusieurs informations à son sujet.
MESURE : 1-1 : Une mesure ne peut exister que si elle concerne un patient.

---------*****----------

ASSOCIATION : CONNAIT

Définition : Associe un service à un patient.

Attributs :

Connectivité :
- SERVICE : 0-N : L'existence d'un service est indépendante du fait qu'il connaît des patients. Il peut en connaître plusieurs.
- PATIENT : 1-N : Un patient n'est considéré que s'il est connu par au moins un service, i.e. le service admission. Il peut être connnu par plusieurs services.

---------*****----------

ASSOCIATION : CH-ELC

Définition : Associe à un champ un élément-code.

Attributs :

Connectivité : CHAMP : 0-N : Un champ peut faire référence à un ou plusieurs éléments-code.
EL-CODE : 1-1: Un élément-code n'existe qu'en relation avec un champ de type code. Cette relation est unique.

---------*****----------

ASSOCIATION : DEFINIT

Définition : Associe une mesure à une définition-de-mesure ou champ.

Attributs :

Connectivité : MESURE : 1-1 : Une mesure ne peut exister sans être en relation avec un champ qui la définit.
CHAMP : 0-N : Un champ peut exister sans être utilisé et peut servir plusieurs fois pour des mesures différentes mais répondant à une même définition.

---------*****----------

ASSOCIATION : DEVELOPPE

Définition : Associe un champ au service qui en est le propriétaire.

Attributs :

Connectivité : SERVICE : 1-N : A un service peuvent être associés des champs. Il existe toujours au moins le champ visite qui est du type 'temps', permettant de mémoriser les dates et heures des vistes.
CODE : 1-1 : Un champ ne peut exister que s'il appartient à un service unique.

---------*****----------

ASSOCIATION : DOSSIER-ST

Définition : Associe un service aux pages qu'il a défini.

Attributs :

Connectivité : SERVICE : 1-N : Un service dès sa création se voit associé une page récapitulative. Il peut en créer plusieurs.
PAGE : 1-1 : Une page est toujours la propriété d'un service unique.

---------*****----------

ASSOCIATION : ELC-PF

Définition : Associe un élément-code à un autre par un lien père-fils, l'ensemble constituant la structure arborescente d'un code.

Attributs :

Connectivité : - père : 0-N ; un élément-code peut ne pas avoir de fils (c'est une feuille) ou bien en avoir plusieurs.
- fils : 0-1 : Dans un arbre, un élément-code fils ne peut l'être que d'un seul élément-code, à l'exception de la racine qui ne possède pas d'ancêtre.

----- °°°°° -----

Contrainte d'intégrité :

1) Dans l'association branche côté fils :
Il n'existe qu'un seul noeud n'ayant aucun père c'est la racine.
2) Dans l'association CH-SERVICE: il n'y a que les champs du service-commun qui peuvent avoir une connectivitè 1-N.

---------*****----------

ASSOCIATION : ENREGISTRE

Définition : Associe une mesure au service qui l'a enregistrée. C'est aussi un lien de propriété.

Attributs :

Connectivité : SERVICE : 0-N : Un service peut exister sans avoir enregistré de mesures concernant un patient. Mais bien souvent, il en enregistrera plusieurs.
PATIENT : 1-1 : Une mesure ne peut être enregistrée que par un service.

---------*****----------

ASSOCIATION : MES-ELC (mesure-élément-code)

Définition : Associe une mesure à un code.

Attributs :

Connectivité : MESURE : 0-N : Une mesure peut ne pas être de type code, mais si c'est le cas il peut faire référence à plusieurs éléments d'un code.

EL-CODE : 0-N: Un élément d'une mesure de type code peut exister indépendamment de toute mesure et plusieurs mesures peuvent y faire référence.

---------*****----------

ASSOCIATION : RASSEMBLE

Définition : Associe un service à un hôpital.

Attributs :

Connectivité : - HOPITAL : 2-N ; Un hôpital a toujours au moins deux services : Le service Admission et le service Commun et au maximum plusieurs).

- SERVICE : 1-1 ; Un service n'existe que s'il appartient à un hôpital.

---------*****----------

Chapitre V: La logique du logiciel apparaissant dans le mode d'emploi

1 Introduction

Dans ce chapitre nous essayerons de découvrir la logique du logiciel au travers de la description de son mode d'emploi qui en est très révélateur. Voilà pourquoi c'est ici que nous décidons de décrire le mode d'emploi.

La compréhension à la lecture du mode d'emploi sera grandement facilitée par l'utilisation en parallèle du logiciel.

2 Les fonctions

2.1 Définitions et remarques

Nous allons présenter les différentes possibilités du logiciel :

Nous verrons d'abord l'identification de l'utilisateur par l'entrée d'un nom de service et de son mot de passe.

Ensuite, l'utilisateur disposera d'une table de travail lui indiquant les cinq grandes rubriques. Ces dernières se décomposent en fonctions et c'est à ce niveau que l'utilisateur travaille sur la base de données.

Chacune des fonctions est subdivisée en :

OBJECTIF : Expliquera de manière succinte la fonction à réaliser.

MODE D'EMPLOI : Description de l'écran qui apparaît et de son utilisation.

----- °°°°° ----- °°°°° -----

TERMINOLOGIE

DATE-MIN : Aucune date ne peut-être antérieure à cette date.

DATE-MAX : Aucune date ne peut-être postérieure à cette date.

DATE DE VISITE : Toute date associée à une mesure pour un patient dans un service.

PREMIERE DATE ACCESSIBLE : C'est la date de visite la plus proche de la date courante. S'il n'existe pas de date de visite correspondant à la date courante, on prendra la date de visite la plus proche dans le futur et à défaut dans le passé.

PAGINATION : Affichage écran par écran d'une liste de données de même type. Pour se déplacer dans la liste, l'utilisateur dispose des touches suivantes :

- Home : Pour aller au début de liste.
- End : Pour aller en fin de liste.
- PgUp : Pour monter d'une page dans la liste.
- PgDn : Pour descendre d'une page dans la liste.

- ‹-, -› : Pour les déplacements latéraux.
- ^, v : pour les déplacements verticaux.

TOUCHES PARTICULIERES : Pour sortir d'une fonction, l'utilisateur peut utiliser deux touches :

Enter : Pour valider le travail effectué dans cette fonction.

Esc : Pour abandonner toutes les modifications réalisées dans cette fonction.

On peut sélectionner une valeur à l'écran, en y positionnant le curseur. Un déplacement latéral ou vertical fait office de sélection de données.

AIDE : A tout moment l'utilisateur peut accéder à un écran d'aide lui indiquant comment réaliser ce qu'il désire.

Pour ce faire il dispose de la touche 'F1' qui provoque l'affichage d'informations sur les finalités des touches utilisables.

IMPRESSION : La touche 'F2' permet l'impression sur papier, de données affichées à l'écran (ex :liste de patients...).

Les pseudo-algorithmes ne tiendront pas compte des touches 'F1' et 'F2', car elles n'apportent rien au niveau des accès nécessaires à la base de données.

----- °°°°° ----- °°°°° -----

Nous avons pris comme option que le clavier utilisé est du même type que celui des PCs d'IBM, ces derniers nous ayant semblés être des bons standards.

2.2 Lecture du mot de passe

OBJECTIF

Pour utiliser le programme, l'utilisateur doit se faire reconnaître à l'aide d'un nom de service et du mot de passe correspondant.

MODE D'EMPLOI

Sur l'écran apparaît un message invitant l'utilisateur à entrer le nom d'un service et le mot de passe correspondant.

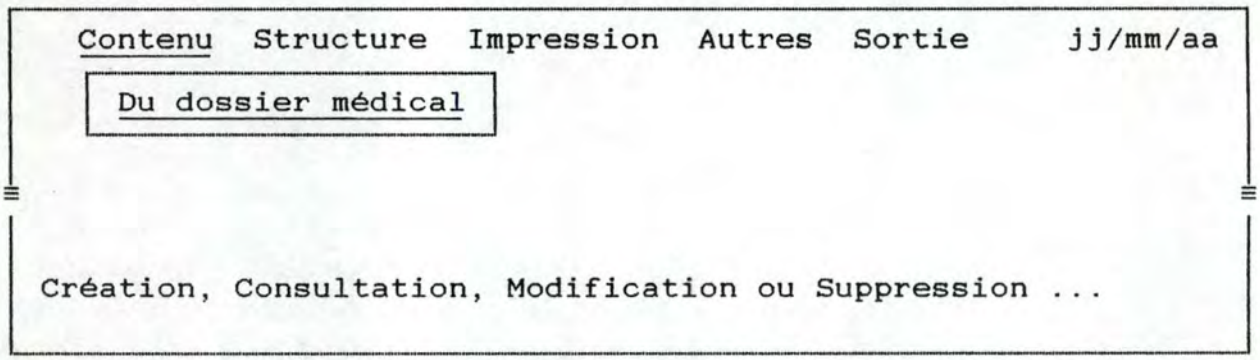
```
NOM DU SERVICE:

MOT DE PASSE :
```

Lors de la frappe du mot de passe, ce dernier n'apparaît pas à l'écran. Après 5 tentatives infructueuses le programme se termine.

2.3 La table de travail

a) Description de la table

Lorsque le mot de passe vient d'être entré l'utilisateur voit apparaître l'écran que nous allons décrire :



L'écran se compose de plusieurs zones :

La 1ère ligne donne la date du jour et les 5 grandes rubriques que nous allons détailler dans les points suivants.

On se déplace entre ces rubriques en employant les flèches gauche et droite.

En-dessous de cette ligne se trouve un menu spécifique à la rubrique courante, permettant d'accéder à des fonctions. On se déplace entre ces fonctions en employant les flèches 'haut' et 'bas'.

La sélection d'une fonction se fait en tapant 'Enter'.

La dernière ligne de l'écran explique de manière succinte ce que l'on peut réaliser avec la fonction courante.

On sort d'une fonction en tapant 'Enter' ou 'Esc' si l'on ne désire pas garder traces des modifications effectuées dans cette fonction.

Remarques : les options et fonctions courantes apparaissent en inversé à l'écran. Sur papier nous simulerons l'inversé par un soulignement.

b) **Les rubriques**

I) **Contenu**

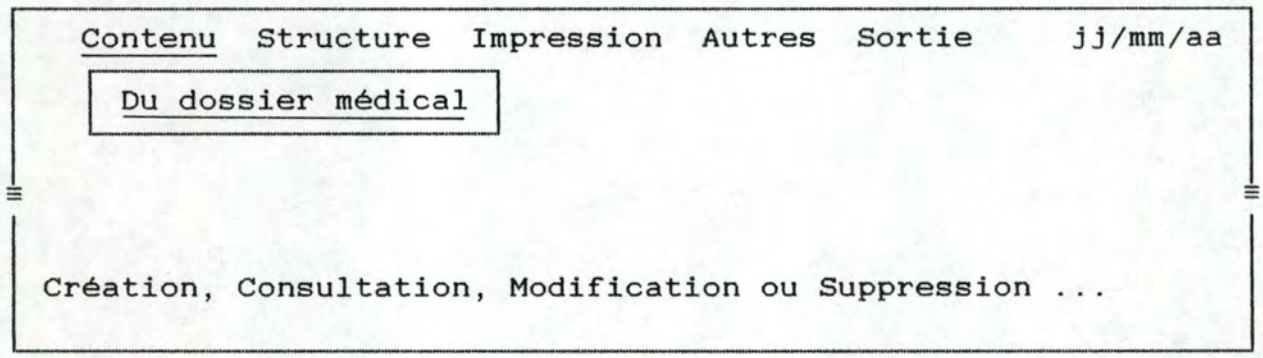
I-1) Le menu

OBJECTIF

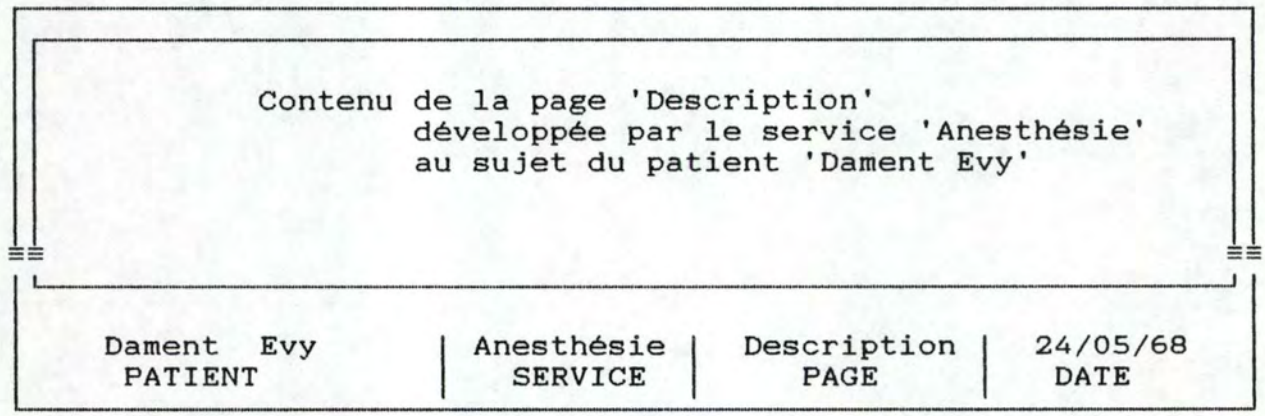
C'est par cette fonction qu'un service pourra compulser la base de données.

MODE D'EMPLOI

L'utilisateur a tapé 'Enter' quand il a vu apparaître l'écran suivant:



Suite à cet écran apparaît :



Par défaut, c'est la première page du dossier médical du premier patient du service courant à la première date accessible.

I-2) Sélection d'une page

a) **Sélection**

OBJECTIF

La sélection d'une page d'un service relatif à un patient et à une date donnée.

MODE D'EMPLOI

L'utilisateur désirant travailler sur les dossiers médicaux doit sélectionner différents paramètres. Il passe dans ce mode en tapant la touche 'Ins', ensuite il lui suffit de choisir un patient, un service, une page et une date.
Pour changer de paramètre on utilise les flèches gauche et droite.

Options par défaut

Si l'on sort et que le patient n'est pas connu du service visité, alors ce sera le premier patient connu de ce service qui sera sélectionné.

Si la page n'est pas connue du service visité, ce sera la première page accessible par le service visiteur qui sera choisie.

Si la date donnée ne correspond à aucune date de visite :
Si le service visiteur peut modifier la page du service visité alors on demande à l'utilisateur s'il désire créer une nouvelle date de visite.
Sinon on prend la première date accessible par le service visiteur dans la page du service visité.

Si le service ne connaît aucun patient ou ne possède aucune page, un écran avertit l'utilisateur de cette situation.

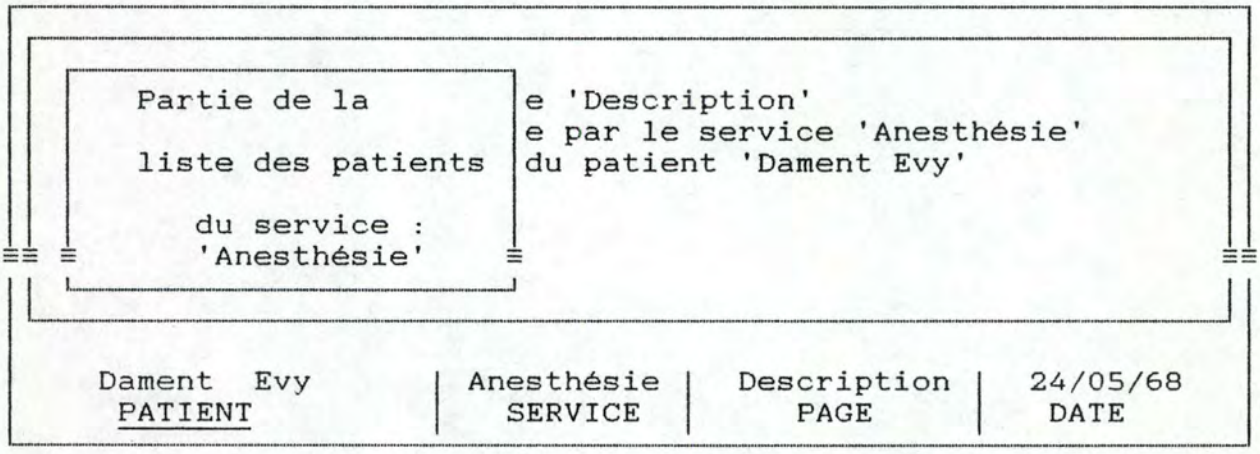
b) **Liste des patients du service visité**

OBJECTIF

L'utilisateur doit identifier un patient.

MODE D'EMPLOI

Après la sélection du paramètre PATIENT l'utilisateur voit apparaître l'écran suivant.



Pour la sélection, l'utilisateur peut :

- Feuilleter une liste d'écrans de patients associées au service courant. Pour chacun d'eux apparaît son nom, son prénom et sa date de naissance. Pour ce faire il utilise les touches de pagination (Home, End ...)

- Donner le nom du patient et dans ce cas il voit apparaître un écran de 'Nom, Prénom & Date-de-naissance' où devrait se trouver ce nom.

Dans les deux cas il lui reste à choisir son patient sur l'écran qui lui apparaît. Ce choix s'effectue soit en tapant 'Enter' qui exprime la fin de cette opération, soit en passant à la sélection d'un autre paramètre.

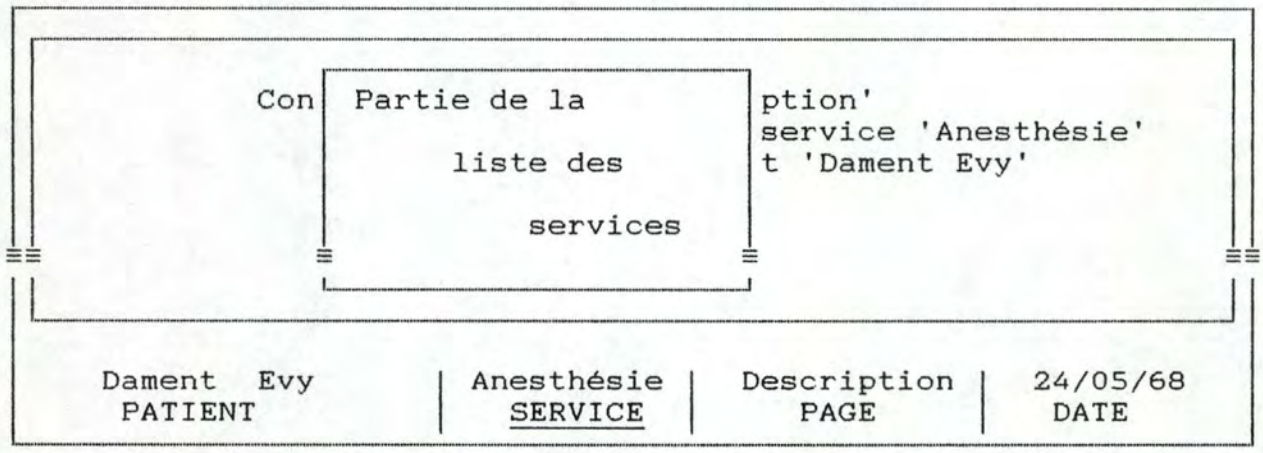
c) **Liste des services**

OBJECTIF

L'utilisateur doit identifier un service.

MODE D'EMPLOI

Lors de la sélection du paramètre 'SERVICE', l'utilisateur voit apparaîte l'écran suivant :



L'utilisateur peut choisir un service en feuilletant une liste d'écrans des services.
Pour ce faire il utilise les touches de pagination (Home, End ...)

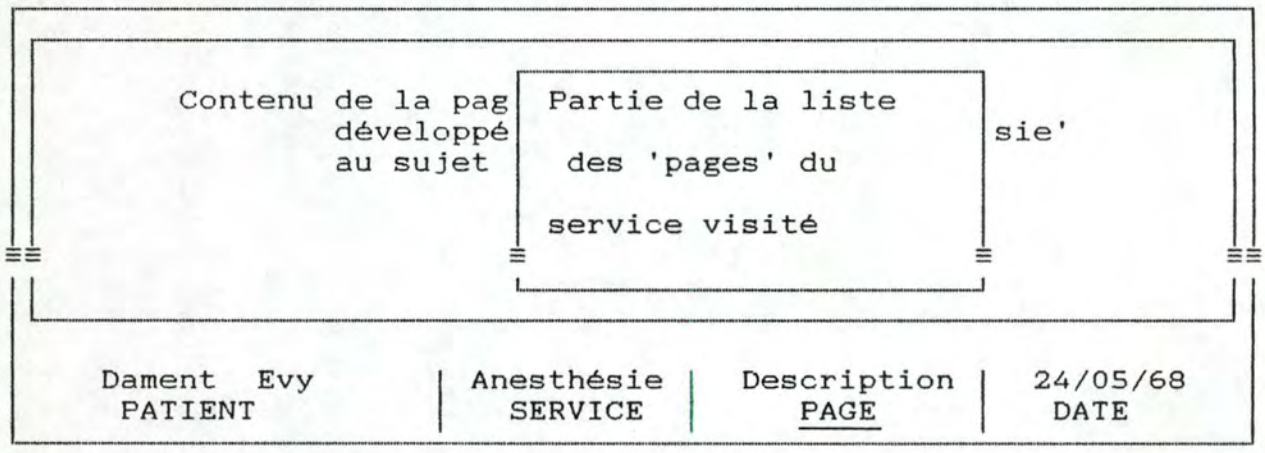
d) **Liste des pages du service visité**

OBJECTIF

L'utilisateur doit identifier une page du dossier médical.

MODE D'EMPLOI

Lors de la sélection du paramètre 'PAGE', l'utilisateur voit apparaître l'écran suivant :



L'utilisateur peut choisir une page du service visité en feuilletant une liste d'écrans de pages associée au service courant.

A chaque page sont associés : un intervalle de temps pour la consultation et un pour la modification.

Seule la liste des pages qui lui sont accessibles au moins en lecture lui sont affichées.

Pour ce faire il utilise les touches de pagination (Home, End ...)

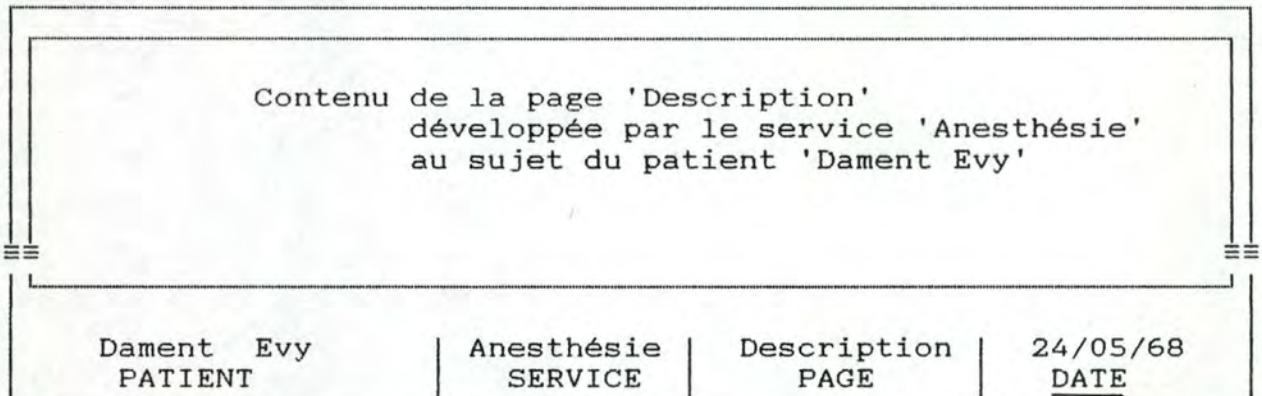
e) **Modification de la date et pagination**

OBJECTIF

L'utilisateur doit choisir une date ou dans le cas d'une page-tableau, un intervalle de temps.

MODE D'EMPLOI

Lors de la sélection du paramètre 'DATE', l'utilisateur voit apparaître l'écran suivant :



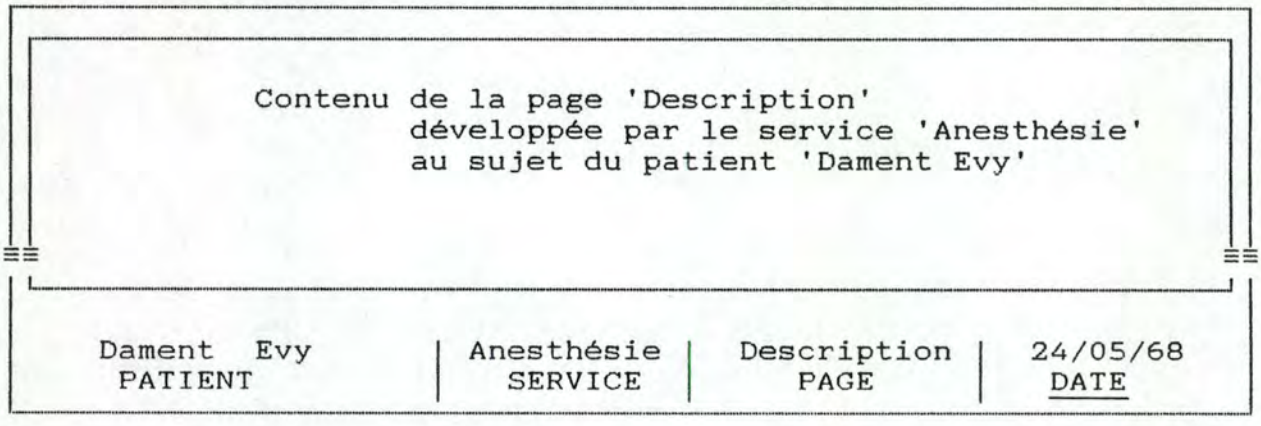
L'utilisateur peut choisir une page du service visité à une date particulière, soit en feuilletant les différents contenus de cette page pour différentes dates. Pour ce faire il utilise les touches de pagination (Home, End ...)
soit en entrant une date directement.

I-3) Actions sur une page

a) **Introduction**

Une fois la page du dossier sélectionnée, l'utilisateur a la possibilité de modifier le contenu de cette page.

Pour ce faire, il dispose de l'écran suivant :



Remarquons que lors de l'accès d'une page à une date qui n'existait pas encore, les différentes mesures correspondantes aux différents champs sont remplis avec leur valeur par défaut.

La page qui vient d'être sélectionnée s'affiche à l'écran, elle peut ne pas être de la même dimension que l'écran physique. L'utilisateur dispose de touches de pagination pour se déplacer dans la page.

b) **Création**

OBJECTIF

Création d'une nouvelle date de visite.

MODE D'EMPLOI

La création du contenu d'une page est soit implicite, c'est le cas où la date de la page (non tableau) correspond déjà à une date de visite et dans ce cas il est possible que déjà une partie des champs contient des mesures. Ce cas revient à modifier le contenu de la page.

Si maintenant l'utilisateur a sélectionné une page pour un patient traité dans un service où il n'existe pas encore de mesure à cette date, alors une nouvelle date de visite est crée et les mesures de la page sont remplies entièrement de leur valeur par défaut. L'utilisateur n'a plus qu'à modifier ces mesures.

Pour créer une nouvelle date de visite : L'utilisateur a sélectionné une page de dossier qu'il a défini, un patient et dans l'option 'DATE' une nouvelle date de visite. Un message dans le bas de l'écran lui annonce que cette date n'existe pas encore et l'invite à valider cette nouvelle date.

c) **Consultation, Modification**

OBJECTIF

L'utilisateur peut soit consulter le contenu d'une page ou modifier son contenu. Ceci n'est possible qu'à la condition que la permission lui soit donnée.

MODE D'EMPLOI

La page affichée peut être consultée ; dans ce cas l'ultilisateur employe des touches pour voyager dans cette page.

Pour modifier le contenu de champs de la page, il suffit de remplacer les anciennes valeurs par les nouvelles en se plaçant sur la mesure à modifier.

d) **Suppression**

OBJECTIF

Suppression du contenu d'un champ.

MODE D'EMPLOI

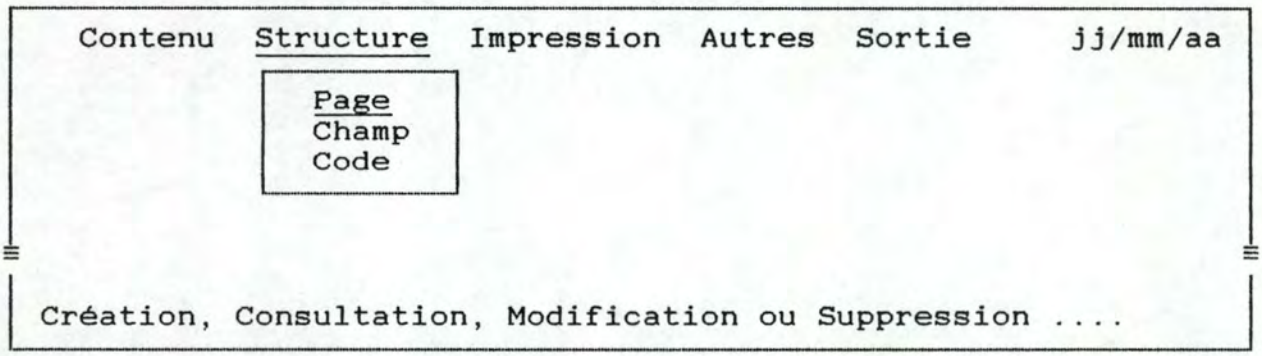
La suppression des mesures ne peut se faire que par l'intermédiaire de la 'page-récapitulative' où il sélectionne les mesures qu'il faut supprimer. Désormais, lors de l'affichage de pages faisant référence à des mesures supprimées on verra apparaître la valeur par défaut des champs correspondants à ces mesures.

Il est toutefois possible de détruire l'ensemble de toutes les mesures ce qui revient à détruire la visite pour le jour courant. C'est la destruction du contenu du champ visité à cette date. Cette destruction est irréversible.

II) **Structure**

II-1) Le menu

Le menu des fonctions pour la rubrique 'Structure' des données apparaît sous la forme suivante :



La ligne du bas de l'écran est différente suivant la sous-fonction sélectionnée.

II-2) Pages

a) **Introduction**

Si l'utilisateur a choisi la sous-fonction 'Page' dans la rubrique 'Structure'. C'est ici que le service crée son dossier médical type.

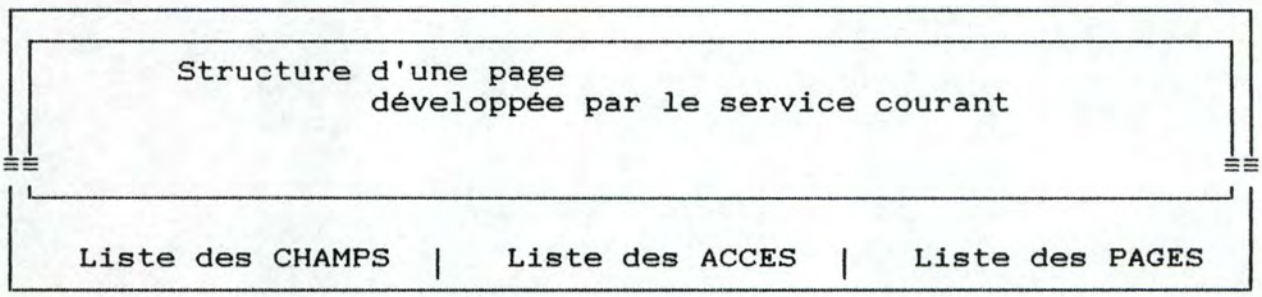
Il peut ajouter des pages à son dossier, en retirer ou modifier la structure d'une page.

OBJECTIF

Edition d'une page-structure.

MODE D'EMPLOI

L'utilisateur voit apparaître l'écran suivant :



La structure d'une page peut-être composée de textes ou de champs.

Pour modifier la Structure d'une page :

'Ins' : Pour ajouter un champ, modifier la liste des accès à cette page ou enfin pour changer de page.

Pour ajouter une ligne de texte dans l'écran, il suffit de se placer à l'endroit désiré grâce aux flèches et de taper la ligne, et de même pour modifier une ligne de texte. Ceci n'est valable que pour les pages non tableau.

Si plusieurs modifications sont apportées à la structure de la page au cours d'une même journée, seule la dernière est considérée.

Dans le cas de pages tableau, la structure de page est remplacée par une liste de noms de champs. Cette liste représente l'ordre dans lequel les champs seront affichés en colonne.

b) **Liste de pages**

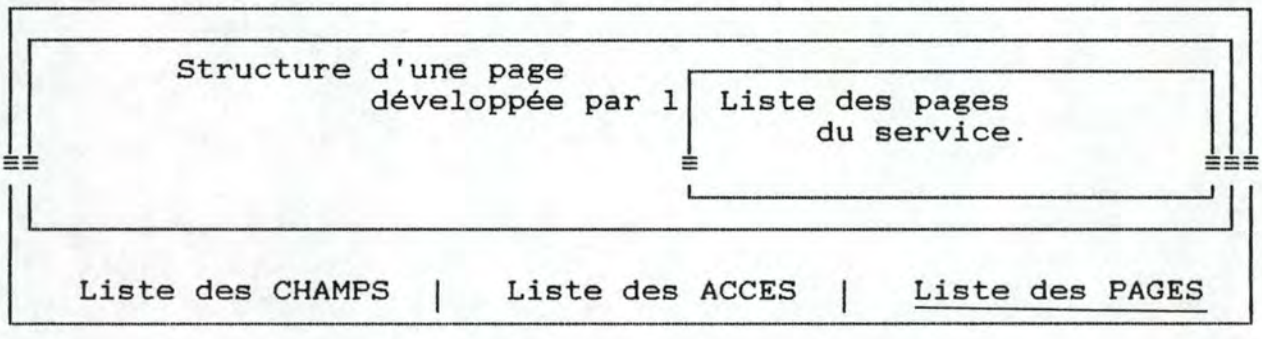
OBJECTIF

Dans le cas où le service désire modifier la structure de son dossier médical, il peut :

- Ajouter des nouvelles pages.
- En retirer.
- En renommer.
- Définir le genre d'une page (normale ou tableau).
- Modifier le statut d'une page.

MODE D'EMPLOI

Toutes ces actions se réalisent autour d'un écran qui affiche la liste des écrans existants :



Il suffit à l'utilisateur de sélectionner une page dans la liste des pages affichées pour pouvoir modifier le nom ou le type de cette page. La sélection s'effectue en employant les touches de pagination, la page sélectionnée est mise en mode inverse :

Ins : Pour insérer une page, un message en bas de l'écran invite l'utilisateur à donner le nom de la nouvelle page. Cette dernière sera automatiquement insérée dans la liste des pages.

Del : Pour retirer une page de la liste il suffit de taper la touche 'Del', un message demande si la page doit être marquée ou être définitivement détruite.

+ : Pour afficher en plus des pages courantes, les pages marquées.

- : Pour sortir de ce mode.

Enter : Pour passer à la structure de la page.

Esc : Pour abandonner les modifications et revenir à la table de travail.

Espace : Pour visualiser le statut de la page et pour pouvoir la modifier : Commun ou non et Marquée ou non.

(Affichage dans le bas de l'écran du statut de la page. La modification du statut de la page s'effectuant à l'aide des flèches et des touches + : mettre et - : retirer)

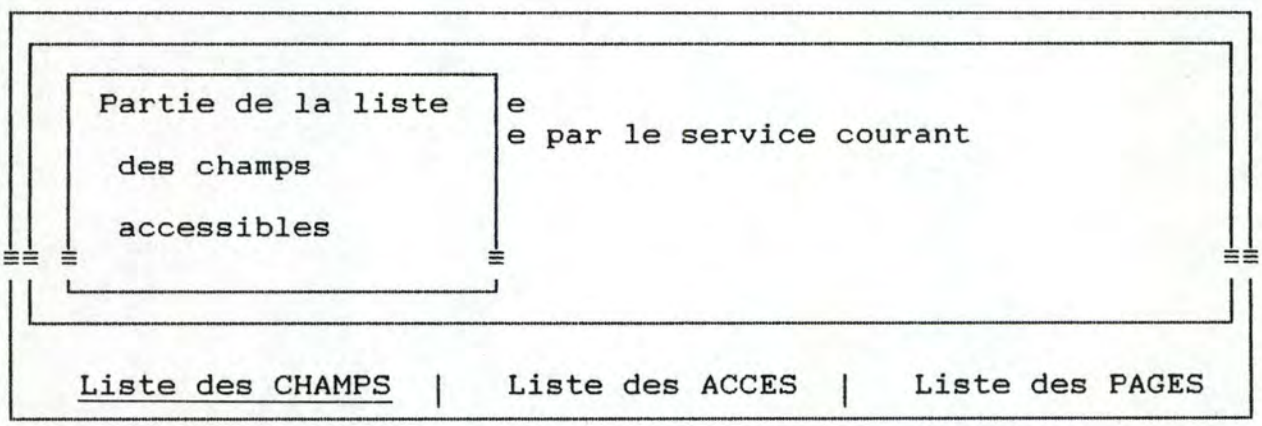
c) **Les champs dans une page**

OBJECTIF

Insertion ou retrait dans la page de champs.

MODE D'EMPLOI

Si l'utilisateur a tapé sur 'Ins' et a choisi la 'liste des champs'.



Pour retirer un champ de la page il suffit de se placer dessus et de taper 'Del'.

Les champs pouvant être intégrés dans une page doivent appartenir au service qui crée la page, il existe toutefois une exception : les champs appartenant au service commun.

Pour l'insertion d'un champ dans une page il suffit de se positionner sur la ligne et de sélectionner le champ.

d) **Accés**

OBJECTIF

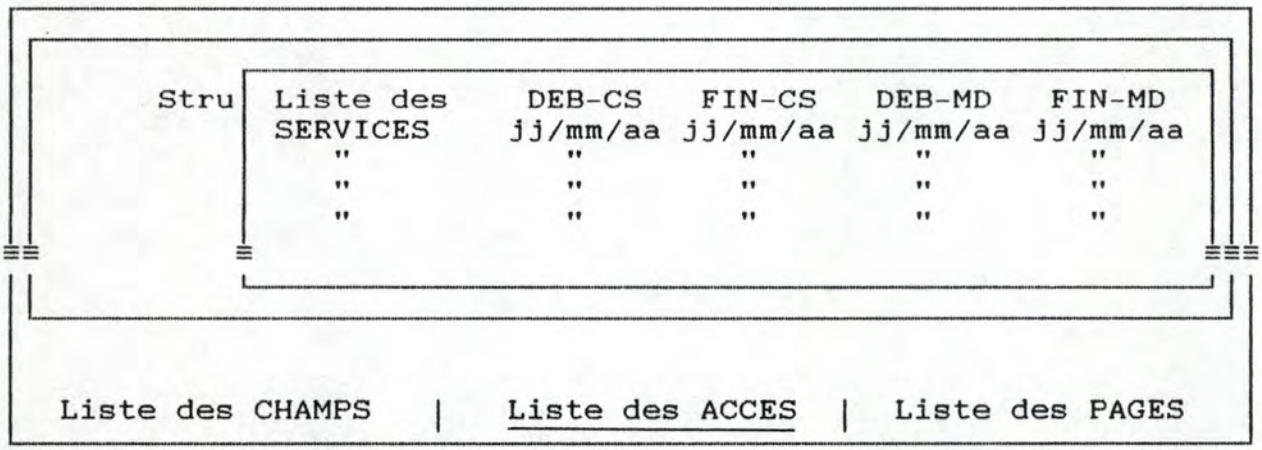
Le service propriétaire décide qu'un certain nombre de services peuvent :

- Modifier le contenu de cette page sur un intervalle de temps.
- Consulter le contenu de cette page sur un intervalle de temps.

Le service propriétaire peut décider de mettre le contenu de cette page dans le service commun.

MODE D'EMPLOI

Si l'utilisateur a tapé sur 'Ins' et a choisi la 'liste des champs'.



DEB-CS = Date à partir de laquelle le service x peut consulter la page (date comprise).
FIN-CS = Date à partir de laquelle le service x ne peut plus consulter la page (date non comprise).
DEB-MD = Date à partir de laquelle le service x peut modifier la page (date comprise).
FIN-MD = Date à partir de laquelle le service x ne peut plus modifier la page (date non comprise).

Les modifications s'effectuant avec les touches de pagination.

II-3) Champs

a) **Introduction**

OBJECTIF

Cette fonction permet à un service de définir un champ.

Pour ce faire, le service dispose d'écrans lui indiquant les différentes variantes possibles. A la fin de la modification, on demande à l'utilisateur si la définition doit avoir ou non des effets rétroactifs. Cette question n'est posée que si l'on n'a pas modifié le type de la donnée.

Il existe plusieurs écrans de définitions (ces derniers sont au nombre de 4 et seront présentés dans les points suivants).

Si la ligne correspond à un choix à réaliser, les différentes alternatives apparaissent, l'utilisateur utilise <- ou -> pour faire son choix.

Si la ligne correspond à une suite de caractères, il suffit de taper cette dernière.

Ins : Permet à l'utilisateur de changer de champ, ce cas est détaillé au point 5.

Remarque : Les lignes n'affichent que la valeur choisie. C'est ainsi que dans la ligne 'Type' on verra par exemple apparaître la valeur 'Entier', le reste des autres alternatives n'est affiché que si la ligne 'Type' devient courante. Pour une raison de clarté nous présentons les écrans et la liste des alternatives possibles pour chaque ligne.

Si plusieurs modifications sont apportées à la structure de la page au cours d'une même journée, seule la dernière est considérée.

b) **Champ : 'Nombre'**

```
.Type          : Entier Réel Suite de caractères Date Heure
.Nom du champ  : .........
.Commentaire   : ........................................
.Commun        : Oui Non
.Genre         : Statique  Dynamique
.Répétitif     : Oui Non
.Nombre Minimum   :0000
.Nombre Maximum        : 9999
.Valeur par défaut     :.9999
.Unité                 : .....
                        Liste Des Champs
```

c) **Champ : 'Suite de caractères'**

```
.Type          : Entier Réel Suite de caractères Date Heure
.Nom du champ  : .........
.Commentaire   : ........................................
.Commun        : Oui Non
.Genre         : Statique  Dynamique
.Répétitif     : Oui Non
.Nombre Minimum         : 0000
.Nombre Maximum         : 9999
.Valeur par défaut      : ................................
                        Liste Des Champs
```

d) **Champ : 'Date'**

```
.Type          : Entier Réel Suite de caractères Date Heure
.Nom du champ  : .........
.Commentaire   : ........................................
.Commun        : Oui Non
.Genre         : Statique  Dynamique
.Répétitif     : Oui Non
.Date  Minimum         : 0000
.Date  Maximum         : 9999
.Date  par défaut      :.9999
.Unité                 : .....
                        Liste Des Champs
```

e) **Champ : 'Heure'**

```
.Type          : Entier Réel Suite de caractères Date Heure
.Nom du champ  : .........
.Commentaire   : ........................................
.Commun        : Oui Non
.Genre         : Statique  Dynamique
.Répétitif     : Oui Non
.Heure Minimum        : 00:00:00
.Heure Maximum        : 24:00:00
.Heure par défaut     : 12:00:00
                        Liste Des Champs
```

f) Liste des champs

OBJECTIF

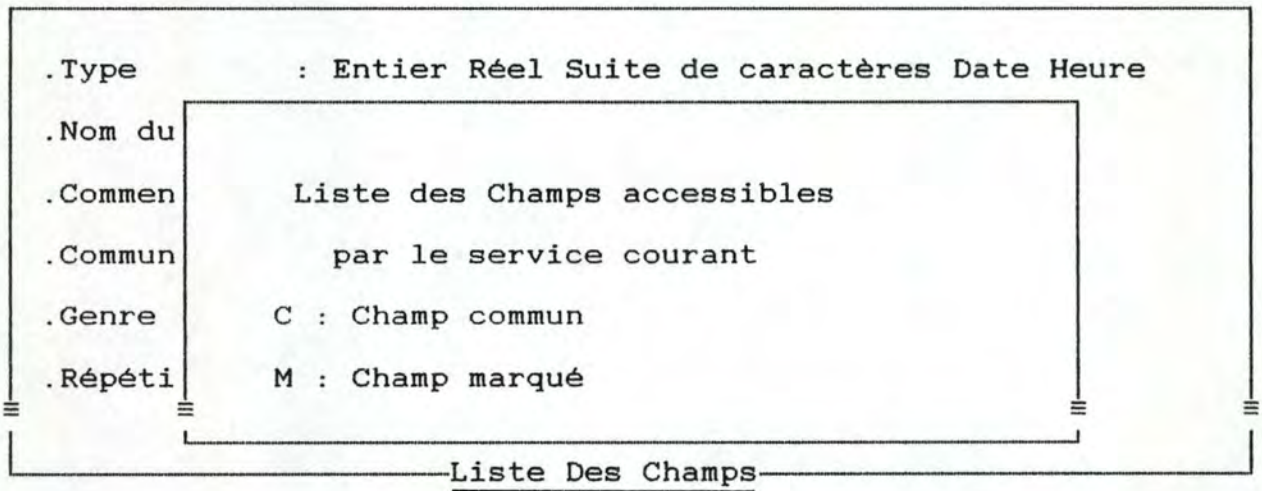
Dans le cas où le service désire modifier la liste des champs accessibles, il peut :

- Ajouter des nouveaux champs.
- En renommer.
- En retirer.

MODE D'EMPLOI

Toutes ces actions se réalisent autour d'un écran qui affiche la liste des champs existant. Les commandes sont toujours les mêmes.

Les champs dont le service n'est pas propriétaire sont précédés d'un 'C' indiquant qu'ils appartiennent au service commun.



Il suffit à l'utilisateur de sélectionner un champ dans la liste des champs affichés pour pouvoir modifier le nom ou le type de ce champ. La sélection s'effectue en utilisant les touches de pagination, le champ sélectionné est mis en mode inverse.

Ins : Pour insérer un champ, un message en bas de l'écran invite l'utilisateur à donner le nom du nouveau champ. Ce dernier sera automatiquement inséré dans la liste des champs.

Del : Pour retirer un champ de la liste il suffit de taper la touche 'Del'. Un message demande si la champ doit être marqué ou être définitivement détruit.

\+ : Pour afficher en plus des champs courants, les champs marqués. Ces derniers sont précédés de la lettre 'M'.

\- : Pour sortir de ce mode.

Enter : Pour passer à la structure du champ.

Esc : Pour abandonner les modifications.

Espace : Pour visualiser le statut du champ et pour pouvoir le modifier : Commun ou non et Marqué ou non.
(Affichage dans le bas de l'écran du statut du champ. La modification du statut du champ s'effectuant à l'aide des flèches et des touche + et -)

Remarquons que la liste de champs est enrichie par les champs appartenant au service commun. Ces champs sont consultables mais non modifiables. Ils peuvent être utilisés dans une page. Leur nom est précédé d'un 'C' indiquant ainsi leur origine.

II-4) Code

a) Introduction

OBJECTIF

C'est dans cette fonction que l'utilisateur va créer, modifier ou supprimer un code. L'utilisateur peut aussi compulser le contenu d'un code qui appartient au service commun et dont il n'est pas le propriétaire.

b) Liste des codes

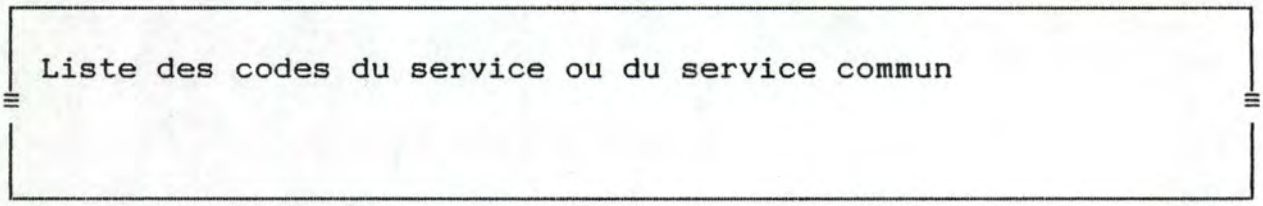
OBJECTIF

Dans le cas où le service désire modifier la liste des codes accessibles, il peut :
- Ajouter des nouveaux codes.
- En renommer.
- En retirer.

MODE D'EMPLOI

Toutes ces actions se réalisent autour d'un écran qui affiche la liste des codes existant. Les commandes sont toujours les mêmes.

Les codes dont le service n'est pas propriétaire sont précédés d'un 'C' indiquant qu'ils appartiennent au service commun.



Il suffit à l'utilisateur de sélectionner un code dans la liste des codes affichés pour pouvoir modifier le nom ou le type de ce code. Ceci n'est possible que si le service est propriétaire du code. La sélection s'effectue en employant les touches de pagination. Le code sélectionné est mis en mode inverse :

Ins : Pour insérer un code, un message en bas de l'écran invite l'utilisateur à donner le nom du nouveau code. Ce dernier sera automatiquement inséré dans la liste des champs.

Del : Pour retirer un code de la liste il suffit de taper la touche 'Del'. Un message demande si le code doit être marqué ou être définitivement détruit.

+ : Pour afficher en plus des codes courants, les codes marqués.
- : Pour sortir de ce mode.

Enter : Pour passer à la structure du code.

Esc : Pour abandonner les modifications.

Espace : Pour visualiser le statut du code et pour pouvoir le modifier : Commun ou non, Marqué ou non, Répétitif ou non...
(Affichage dans le bas de l'écran du statut du code. La modification du statut du code s'effectuant à l'aide des flèches et des touche + et -)

Remarquons que la liste de codes est enrichie par les codes de type code appartenant au service commun. Ces codes sont consultables mais non modifiables, ils sont précédés par la lettre 'C'.

c) Modification d'un code

OBJECTIF

Compléter un libellé, par la création de nouveaux libellés, supprimer des libellés ou modifier des libellés.

MODE D'EMPLOI

Pour travailler l'utilisateur dispose de l'écran suivant :

```
Liste arborescente des libellés du code

      Les libellés sont présentés de manière arborescente :

       Niveau 1-1
        Niveau 2-1
          Niveau 3-1
          Niveau 3-2
       Niveau 1-2
```

Pour la sélection l'utilisateur dispose des touches usuelles de pagination.

Ins : Insertion d'un nouveau libellé.
Del : Pour la destruction d'un libellé.

F10 : Pour développer ou non un branche.
Exemple : Si l'on se place sur la ligne 'Niveau 1-1' et que l'on tape sur 'F10', les lignes 2-1, 3-1 disparaîtront. Pour les faire réapparaître il suffit de retaper sur 'F10'.

Pour modifier un libellé il suffit de se placer sur la ligne où il se trouve et de modifier la ligne avec les commandes usuelles.

III) **Impression**

III-1) Le menu

C'est par cette rubrique que l'utilisateur peut avoir une trace sur papier de la base de données.

Il dispose des fontions suivantes :

```
  Contenu  Structure  Impression  Autres  Sortie      jj/mm/aa

                      +-------------------------+
                      | Longueur d'une page {xx}|
                      +-------------------------+
                      | Liste des patients      |
                      | Liste des services      |
                      +-------------------------+
                      | Liste des pages         |
                      | Liste des champs        |
                      +-------------------------+

Impression de données sur papier.
```

En cas de sortie de liste de données (patients,services...), l'utilisateur voit apparaître la liste de toutes les données disponibles.

Les éléments sélectionnés apparaissent en inversé à l'écran.

F10 : Sélection de tous les éléments de la liste. Pour désélectionner il suffit de retaper sur cette touche.

Esc : abandon de l'impression.

Space : Pour sélectionner ou désélectionner un élément de la liste définitive.

\+ : Pour visionner les éléments marqués.
\- : Pour sortir de ce mode de visualisation.

Enter : Pour commencer l'impression de la liste des éléments sélectionnés.

III-2) Les Fonctions

a) **Longueur d'une page**

OBJECTIF

L'utilisateur ayant choisi cette fonction, peut modifier le nombre de lignes par page.

Dans le bas de l'écran apparaît :

```
┌──────────────────────────────────────────────────────────────────┐
│ Longueur d'une page : x                                          │
└──────────────────────────────────────────────────────────────────┘
```

x étant l'ancienne valeur. Il suffit de rentrer la nouvelle valeur.

b) **Liste des patients**

OBJECTIF

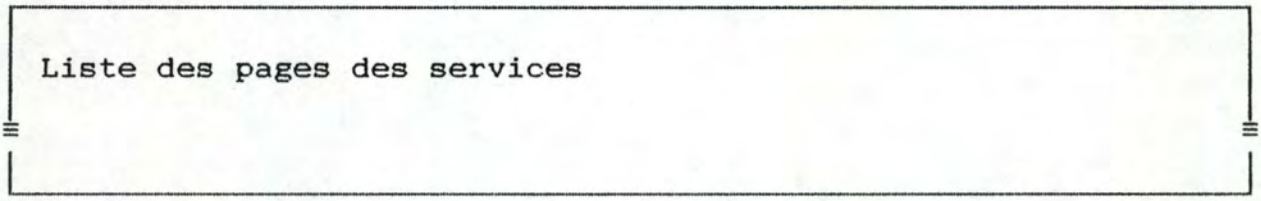
La sélection de cette fonction permet la sortie sur imprimante de la liste totale ou partielle des patients connus par le service.

```
┌──────────────────────────────────────────────────────────────────┐
│ Liste des patients du service courant                            │
≡                                                                  ≡
└──────────────────────────────────────────────────────────────────┘
```

c) **Liste des services**

OBJECTIF

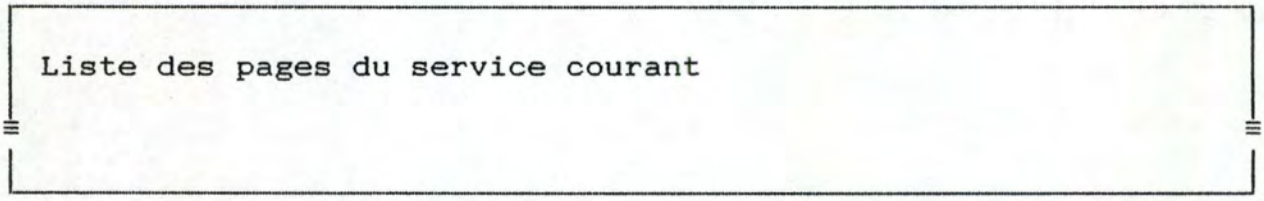
La sélection de cette fonction permet la sortie sur imprimante de la liste totale ou partielle des services de l'hôpital.



d) **Liste des pages**

OBJECTIF

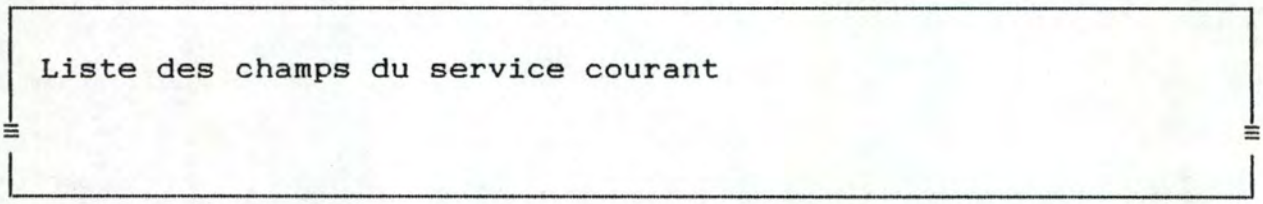
La sélection de cette fonction permet la sortie sur imprimante de la liste totale ou partielle des pages définies par le service.



e) **Liste des champs**

OBJECTIF

La sélection de cette fonction permet la sortie sur imprimante de la liste totale ou partielle des champs connus par le service.

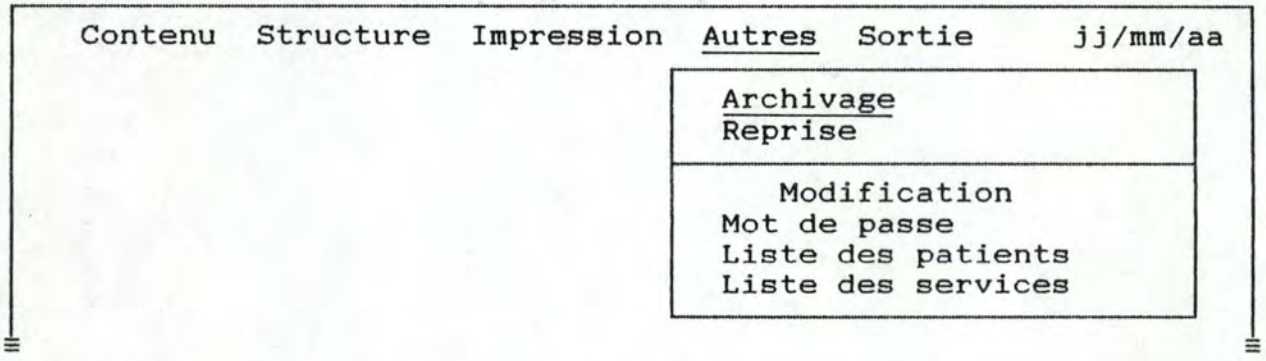


IV) **Divers**

IV-1) Le menu

Cette rubrique permet à l'utilisateur d'employer des fonctions plus particulières.

Il dispose des fontions suivantes :



Notons que la fonction 'Modification Liste des services' n'est accessible qu'à partir de l'HOPITAL.

IV-2) Les Fonctions

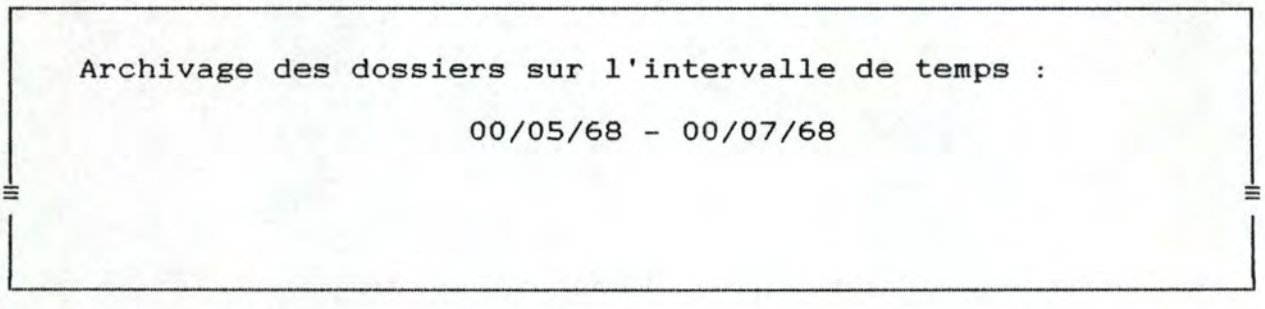
a) **Archivage**

OBJECTIF

L'archivage des données se réalise sur l'ensemble de la base de données pour des intervalles de temps ayant comme unité de temps le mois. Cela revient à mémoriser toutes les données nécessaires au service pour reconstituer l'ensemble des dossiers médicaux dans l'espace de temps archivé : stocker les mesures et les structures nécessaires à la lecture de ces mesures.

MODE D'EMPLOI

L'utilisateur est invité à rentrer l'intervalle de temps sur lequel s'effectuera l'archivage.



Une fois la date rentrée l'utilisateur est invité à mettre des disquettes préformatées dans son lecteur de disquettes.

Une fois l'archivage réalisé l'utilisateur peut s'il le désire retirer de sa base de données celles qui viennent d'être archivées.

```
|                                                          |
|        Voulez-vous retirer ces données ? O/N             |
|                                                          |
```

b) **Reprise**

OBJECTIF

La reprise permet à l'utilisateur de reprendre des données qui ne se retrouvent plus actuellement dans la base de données.

MODE D'EMPLOI

Dans un premier temps l'utilisateur est invité à introduire la première disquette archive.

```
┌──────────────────────────────────────────────────────────┐
|                                                          |
|   Désarchivage de dossiers :                             |
|                                                          |
|        Voulez-vous Insérer la première disquette         |
|                                                          |
└──────────────────────────────────────────────────────────┘
```

Une fois la disquette entrée taper 'Enter'. L'intervalle de temps archivé sur la disquette et le numéro de la disquette apparaissent à l'écran.

```
≡                                                          ≡
|  Disquette N° : 2          Intervalle : 00/05/68 - 00/07/68  |
≡                                                          ≡
```

Si le numéro de la disquette n'est pas le numéro 1, l'utilisateur est invité à entrer la première disquette.

Ensuite il suffit de présenter les disquettes par numéro croissant.

Remarque : A tout moment l'utilisateur peut interrompre la reprise de données en tapant 'Esc'. Les données précédemment lues seront perdues.

c) **Modification du mot de passe**

OBJECTIF

La modification du mot de passe.

MODE D'EMPLOI

a) Sur l'écran un message invite l'utilisateur à taper l'ancien mot de passe.

Une fois l'ancien mot de passe reconnu, l'utilisateur est invité à taper deux fois le nouveau mot de passe.

Les mots de passe tapés au clavier n'apparaissent pas à l'écran, sécurité oblige.

d) **Modification de la liste des patients**

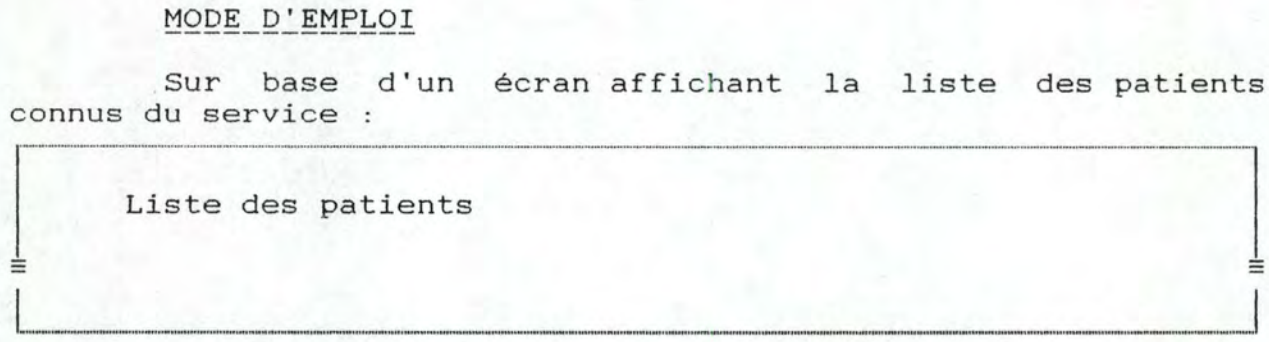
OBJECTIF

Tous les services peuvent ajouter un patient à la liste des patients. Un patient ne sera en fait accepté que s'il n'existe pas encore de patient ayant le même nom, prénom et date de naissance.

La modification d'un patient, i.e. la modification d'un des trois éléments constituant l'identifiant d'un patient ne peut se faire que par le service 'admission'.

MODE D'EMPLOI

Sur base d'un écran affichant la liste des patients connus du service :



L'utilisateur peut ajouter ou retirer un patient de la liste des patients qu'il connaît.

Ins : Pour ajouter un nouveau patient.

Del : pour retirer de la liste le patient courant.

La sélection du patient s'effectuant à l'aide des touches habituelles (Home, End, <-, ->...)

e) **Modification de la liste des services**

OBJECTIF

L'ajout d'un service.

MODE D'EMPLOI

L'hôpital peut ajouter ou retirer un service de la liste des services.

Sur base d'un écran affichant la liste des services :

Ins : Pour ajouter un nouveau service.

Del : pour retirer de la liste.

La sélection du patient s'effectuant à l'aide des touches habituelles (Home, End, ‹-, -›...)

V) **Sortie**

OBJECTIF

Clôturer l'ensemble des actions réalisées lors de la session de travail.

2.4 Journal

Toutes les modifications de la base de données doivent être enregistrées dans un ensemble de fichiers composant un journal. Le but de ce journal est de pouvoir retracer la vie de la base de données. Ce journal pourra donc servir de preuve dans certains cas. Pour cette raison, il est nécessaire que le journal ne soit accessible que par un très petit nombre de personnes.

Le journal pourra aussi servir à recréer la base de données en cas de destruction totale ou partielle.

Chapitre VI : Les fonctions et la base de données

Introduction

Nous allons décrire dans ce chapitre la méthode utilisée pour définir une base de données à partir d'informations exprimées dans le modèle "Entité-Association".

Chaque fonction développée dans le chapitre précédant ne travaille que sur une partie de la base de données. Nous obtiendrons une description complète de cette base en réunissant les résultats obtenus par l'application de la méthode pour chacune de ces parties.

En fait, nous n'employerons cette méthode que pour les fonctions les plus simples, pour les autres nous expliciterons en pseudo-langage les accès nécessaires. Pour chacune des fonctions, nous compléterons la description par un sous-graphe de la partie de la base de données utilisée.

1 Le modèle des accès possibles

a) Introduction

Le Modèle des Accès Possibles permet de définir une base de données sans être tributaire d'une implémentation.

Pour comprendre ce qui suit, il est nécessaire que l'utilisateur maîtrise ce qui est développé dans [HAINAUT].

Nous allons donner ici, un exemple simple introduisant les différents concepts définis dans [HAINAUT].

a) Soit : Un client pouvant passer des commandes, chacune portant sur un produit et au maximum une commande par jour.

Nous modéliserons cela comme suit dans un modèle Entité-Association:

Les Entités

CLIENT : Toute personne pouvant passer une commande.
- NUM-CLI : Le numéro du client. (Identifiant)
- NOM-CLI : Le nom du client.

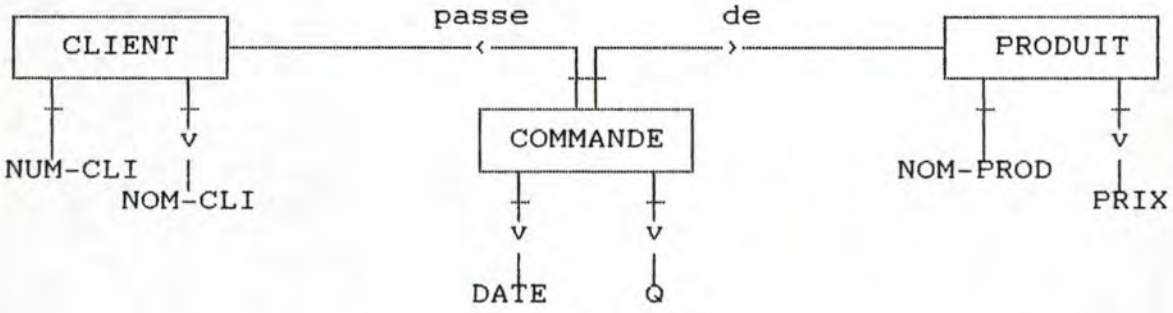
PRODUIT : Tout produit qui peut faire l'objet d'une commande.
- NOM-PROD : Le nom du produit. (Identifiant)
- PRIX : Le prix du produit.

Association

COMMANDE : Représente une commande d'un produit par un client.
- DATE : La date où la commande a été passée.
- Q : La quantité du produit commandée.

Connectivités

Passe : 0-N : Un client peut exister sans passer de commande et peut en passer plusieurs.
De : 0-N : Un produit peut exister sans qu'on ne le commande et il peut être commandé par plusieurs utilisateurs.
Le schéma des accès possibles est le suivant :



Nous avons trois **articles** qui posssèdent chacun deux **items**. L'article CLIENT possède les items NUM-CLI et NOM-CLI.

Les liens qui relient les articles entre eux, sont appelés **chemins**. Nous avons les chemins 'passe' et 'de'. Il est à souligner qu'il existe un chemin par direction. C'est ainsi que le chemin 'passe' peut-être décomposé en 'cli-com' qui va de CLIENT vers COMMANDE et en 'com-cli' qui va de COMMANDE vers CLIENT.

Les items sont reliés aux articles par une ligne. La ligne transversale indique que l'item doit obligatoirement exister. Le même symbolisme est employé sur les chemins 'passe' et 'de', du côté COMMANDE, indiquant qu'une commande ne peut exister sans être reliée à un produit et à un client.

Le symbole '<' sur le chemin 'passe', indique qu'à un client peut correspondre plusieurs commandes(c'est un chemin de type 'one to many'). Le même symbolisme est employé pour indiquer qu'à un PRIX peut correspondre plusieurs PRODUITs.

------ °°°°°° ------

Pour déterminer les chemins nécessaires nous devons montrer les actions que l'on désire réaliser sur la base de données. Pour ce faire nous utilisons le langage ADL.

Exemples :
Création d'un client 'BOND' dont le numéro est 007:

```
create CLIENT((:NOM-CLI = 'BOND') and (:NUM-CLI = 007))
```

Création d'une commande passée par 007 sur le produit 'BERETTA'.

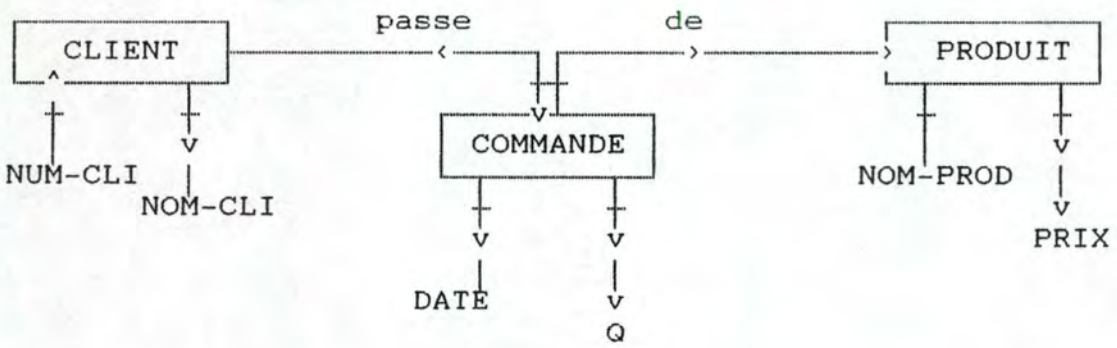
```
create COMMANDE((cli-com:CLIENT(:NUM-CLI = 007)
        and  (pro-com:PRODUIT(:NOM-PROD = 'BERETTA')
        and  (:Q = 1)
        and  (: date = 14/05/1954))
```

On peut aussi utiliser les primitives :'modify', 'delete', 'first', 'next', 'prev', 'last'.

Pour la consultation :
Je désire connaître la somme dépensée en commande par le client 007. Ce qui revient à passer en revue toutes les commandes de 007.

```
{J'associe à la variable 'Xcli' le client 007}
Xcli :=  CLIENT(:NUM-CLI = 007);

depense := 0;

for Xc := COMMANDE((cli-com:cli)) do{toutes les commandes de 007}
        Xp := PRODUIT(com-prod : Xc);
        depense := depense + PRIX(:Xp) * Q(:Xc);
endfor;

print (depense); { affichage du résultat }
```

Le schéma des accès nécessaires pour la consultation que l'on vient de décrire :



-On voit dans ce schéma que les chemins utilisés sont : cli-com et com-prod.

-Et que l'on utilise pas les attributs 'NOM-CLI', 'DATE' et 'NOM-PROD'.

Le sens des nouvelles flèches indiquant le sens de la lecture.

Pour les items :
On lit le PRIX d'un produit à partir du PRODUIT, on identifie un client à partir de 'NUM-CLI'.

Pour les articles:
On choisit une commande à partir d'un client et on choisit un produit à partir d'une commande.

Modèle des Accès Possibles : *HOPIDOS*

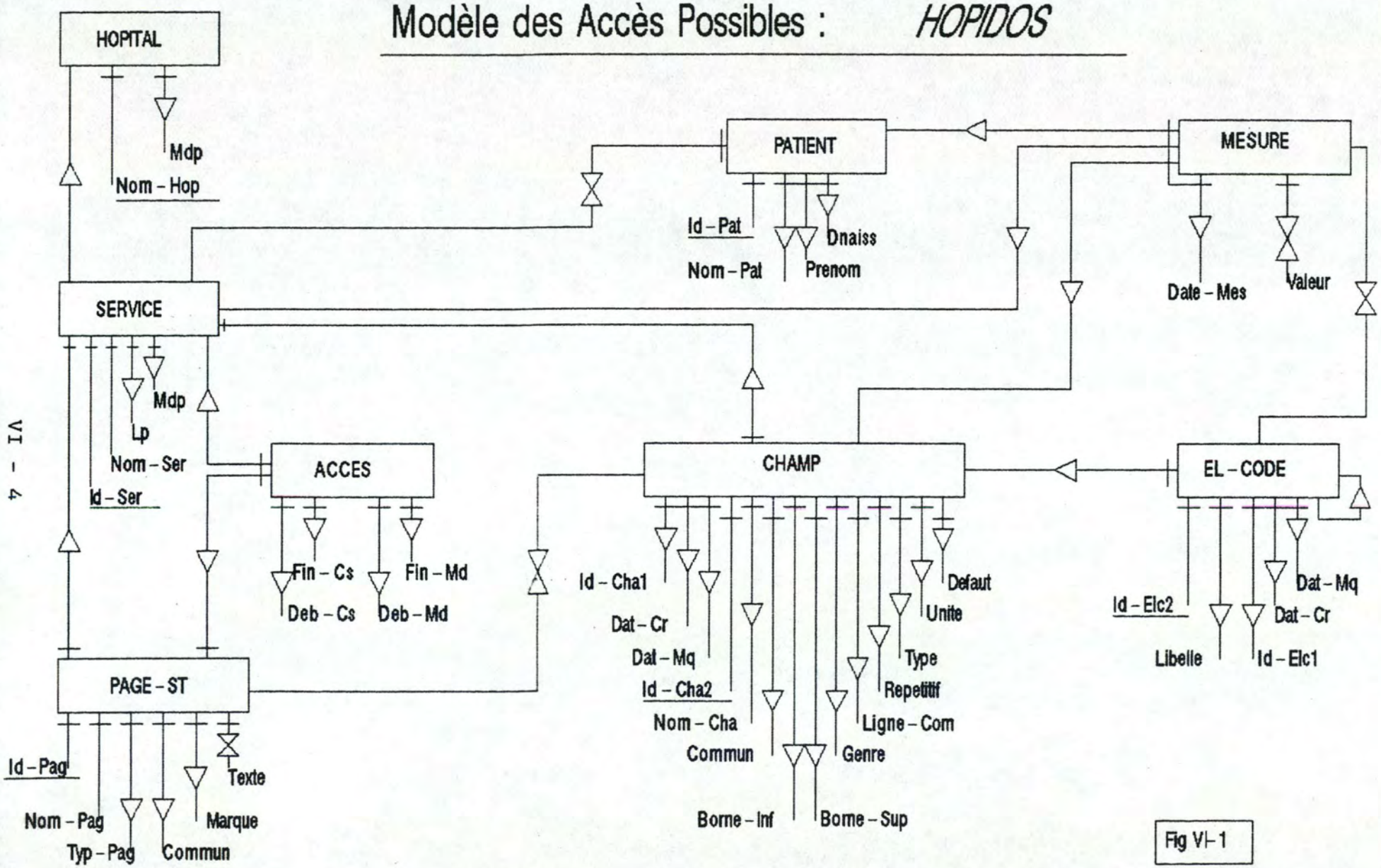


Fig VI-1

2 Les fonctions et la BD

2.1 Définitions et remarques

En développant brièvement les fonctions, nous allons pouvoir déterminer les accès nécessaires à la base de données.

Nous ne détaillerons que la fonction "Mot de passe" à titre d'exemple. Pour le reste des fonctions nous ne fournirons qu'un algorithme succinct, en nous contentant des instructions clefs révélant le chemin suivi pour accéder aux données.

Pour les fonctions trop complexes, nous nous sommes limités à donner les identifiants clefs, les chemins sont fournis par l'intermédiaire du sous-graphe de la base de données utilisée par cette fonction.

Si cette solution manque de rigueur, elle a l'avantage d'être claire. En effet, bien vite nous risquons d'obtenir des algorithmes ADL très lourds, indéchiffrables et sources d'erreurs.

Il existe enfin des fonctions que nous ne développerons pas du tout, du fait que ces dernières n'employent pas de nouveaux chemins. C'est le cas de l'impression de listes, l'archivage, le désarchivage...

----- °°°°° ----- °°°°° -----

Pour ne pas dérouter le lecteur nous reprendrons le même plan que celui du chapitre précédent.

Chacune des fonctions est subdivisée en :

SPECIFICATION : Se décompose en :

Entrées : Les paramètres qui doivent être fournis à cette fonction.

Préconditions : Les conditions que doivent respecter les paramètres entrés.

Sorties : Les paramètres fournis en sortie de cette fonction.

Postconditions : Les conditions que doivent respecter ces paramètres en sorties.

Nous supposerons que chaque fonction pour son bon fonctionnement a besoin d'une base de données cohérentes, i.e. respectant les contraintes d'intégrités. Et de même nous supposerons que toutes les fonctions laissent une base de données cohérente à leur sortie.

ALGORITHME : L'algorithme de cette fonction. Bien souvent nous nous contenterons d'un pseudo-algorithme écrit en français.

ACCES_NECESSAIRES_: Un schéma des accès nécessaires à la réalisation de la fonction.

----- °°°°° ----- °°°°° -----

Les pseudo-algorithmes ne tiendront pas compte des touches 'F1' et 'F2', car elles n'apportent rien au niveau des accès nécessaires à la base de données.

----- °°°°° ----- °°°°° -----

NOTATION

- Actions Possibles :(Entre parenthèses les noms en ADL)

CS : Consultation. CR : Création.
MD : Modification. DT : Destruction.

DB : (Début)Premier article.(FIRST)
DN : Dernier article. (LAST)
SU : Article Suivant. (NEXT)
PR : Article Précédant. (PREV)
TR : Trouver (FIND)
Commentaires : Recherche d'un article sur base d'une clef. A cette clef doit correspondre un article.

CH : Chercher (SEARCH)
Commentaires : Chercher un article sur base d'une clef. A cette clef ne doit pas correspondre nécessairement un article. Dans ce cas, cette action rend l'article qui possède une clef plus grande ou égale à la clef fournie. Il existe un statut indiquant si la clef de recherche correspond à celle associée à l'article trouvé.

- notation des articles.

Acc : ACCES. Cha : CHAMP. Elc : EL-CODE.
Hop : HOPITAL. Pag : PAGE. Pat : PATIENT.
Mes : MESURE. Ser : SERVICE.

-Les variables représentant des occurrences d'articles commenceront par la lettre X.

Exemple : XId_Pat : Indique une valeur d'identifiant interne d'un patient.

2.2 Lecture_du_mot_de_passe

SPECIFICATION_: MotDePasse

Entrée_: /

Précondition_: /

Sorties_:

XId-Ser : L'identifiant d'un service.

OK : - True : Le service et le mot de passe rentrés ont été reconnus comme valides.
- False : Après 5 tentatives l'utilisateur n'a pu rentrer un couple valide { nom de service , mot de passe}.

Postcondition :

Si OK est 'TRUE', XId-Ser est l'identifiant d'un service existant.

ALGORITHME

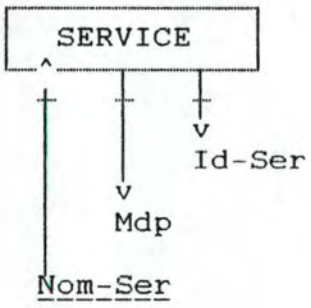
Algorithme prédicatif :

```
I := 1; OK := False;
WHILE (not OK) and ( I<= 5 ) DO
  INPUT (XNom-Ser, XMdp);

  Xser := Service((:Nom-Ser = XNom-Ser) AND
                  (:Mdp = XMdp));

  IF Xser THEN OK := TRUE;
               XId-Ser := Id-Ser(:Xser);
  ENDIF;
  I := I + 1;
ENDWHILE;
```

Algorithme effectif :

```
I := 1; OK := False;

WHILE (not OK) and ( I<= 5 ) DO

        INPUT (XNom-Ser, XMdp);

        Xser := Service(:Nom-Ser = XNom-Ser);

        IF Xser THEN
         IF XMdp = Mdp(:XSer) THEN
            OK := TRUE;
            XId-Ser := Id-Ser(:Xser);
         ENDIF
        ENDIF;

        I := I + 1;

ENDWHILE;
```

ACCES NECESSAIRES



2.3 La table de travail et ses rubriques

I) **Contenu**

I-1) Le menu

SPECIFICATIONS : Contenu

Entrées :

XId-Ser : L'identifiant du service de l'utilisateur.

XId-SerV : Identifiant du service visité par l'utilisateur.

XId-Pag : Identifiant d'une page du service XId-SerV.

Date-Choix : Date à laquelle s'effectue l'affichage du contenu.

Préconditions :

XId-Ser, XId-SerV : doivent être des identifiants de services existants.

XId-Pag : doit être l'identifiant d'une page du service XId-SerV.

Date-Choix : DATE-MIN <= Date-Choix <= DATE-MAX.

Sorties :

XId-SerV : Identifiant du service visité par l'utilisateur.

XId-Pag : Identifiant d'une page du service XId-SerV.

Date-Choix : Date à laquelle s'effectue l'affichage du contenu.

Postconditions :

XId-Ser, XId-SerV : doivent être des identifiants de services existants.

XId-Pag : doit être l'identifiant d'une page du service XId-SerV.

```
 Date-Choix : DATE-MIN <= Date-Choix <= DATE-MAX.
```

ALGORITHME

```
fin := false;

{ Affichage du contenu de la page courante }
Affiche-Pag(XId-SerV,XId-Pag,XId-Pat);

WHILE  NOT fin DO

{Consultation, modification... du contenu de la page courante }
        Action-Pag(XId-Ser,XId-SerV,XId-Pat,Date-Choix,touche);

        CASE touche OF
{Modification des paramètres de sélection }
           Ins : Selection(XId-Ser,XId-SerV,XId-Pat,Date-Choix);

           F1, F2 ...

{Fin de la fonction sur les contenus }
           Esc,Enter : fin := TRUE;

        ENDCASE;
ENDWHILE;
```

APPEL

- Affiche-Pag; cfr 2.3.b.I-3
- Action-Pag; cfr I-3
- Selection; cfr I-2

I-2) Sélection d'une page

a) **Sélection**

SPECIFICATION : SELECTION

Remarque : Le choix des paramètres par défaut s'effectue dans les fonctions Select-...

Entrées :

XId-Ser : Identifiant du service qui recherche des données.

XId-SerV : Identifiant du service dont on utilise les pages.

XId-Pag : L'identifiant d'une page du service identifié par XId-SerV.

XId-Pat : L'identifiant d'un patient.

Choix-Date : Une date sur laquelle s'effectuera la recherche.

Préconditions :

XId-Ser, XId-Ser-V correspondent à des services existants.

XId-Pag doit être l'identifiant d'une page du service identifié par XId-SerV et accessible au moins en consultation par le service identifié par XId-Ser.

XId-Pat : doit correspondre à un patient connu du service identifié par XId-Ser-V.

CHOIX-DATE : DATE-MIN <= CHOIX-DATE <= DATE-MAX.

ALGORITHME

```
choix := ChoixPat;
fin := FALSE;

WHILE NOT fin do

  CASE choix of
        ChoixPat : Select-Pat(XId-SerV,XId-Pat,choix,touche);

        ChoixSer : Select-Ser(XId-Ser,XId-SerV,choix,touche);

        ChoixDat : Select-Pag(XId-Ser,XId-SerV,XDate,choix,touche);

        ChoixPag : Select-Dat(XId-Ser,XId-SerV,XId-Pag,choix,touche);
  ENDCASE;
  IF touche  = Esc OR touche = Enter THEN fin := TRUE ENDIF;

ENDWHILE;
```

APPEL

- Select-Pat;
- Select-Ser;
- Select-Pag;
- Select-Dat;

b) **Liste des patients du service visité**

SPECIFICATION : Select-pat:

Entrées :

XId-Ser : L'identifiant du service connaissant les patients.

XId-Pat : L'identifiant d'un patient.

Préconditions :

XId-Ser doit correspondre à un service existant.

Sorties :

XId-Pat : l'identifiant d'un patient.

choix : Le paramètre suivant sélectionné.

touche : La dernière touche enfoncée.

Postconditions :

XId-Pat : L'identifiant d'un patient 'connu' du service identifié par Id-Ser.

choix ∈ {ChoixDat,ChoixPat,ChoixSer}

touche ∈ {gauche, droite, Esc , Enter}

Si touche = Esc ou Enter alors choix = ChoixPat.

ALGORITHME

Afficher une page de patients dont le premier est XId-Pat

Répéter

Pagination sur les pages de patients.

ELSE INPUT(XNom,XPrenom,XDnaiss);
Afficher la page de Patients dont le nom, prénom & date de naissance sont plus grands ou égaux au nom proposé.

jusqu'à la frappe d'un code de fin cfr RETOUR.

ACCES NECESSAIRES

A l'initialisation :

XPat := PATIENT(:Id-Pat = XId-Pat);

Pour la pagination :

XPat := FIRST/LAST/PREV/NEXT PATIENT [
XPat(SER-PAT : SERVICE (:Id-SER =XId-Ser))
SORTED (Nom-Pat,Prenom,Dnaiss)];

Pour la recherche sur base du nom :

```
XPat := NEXT PATIENT (SER-PAT : SERVICE (:Id-SER = XId-Ser))
              AND (:NOM >= XNOM)
              AND (:PRENOM >= XPrenom)
              AND (:DNAISS >= XDNAISS)
              SORTED (Nom-Pat,Prenom,Dnaiss);
```

c) **Liste des services**

SPECIFICATION : Select-Ser

Entrée :

XId-SerV : l'identifiant d'un service.

Précondition :

XId-Ser-V : L'identifiant d'un service qui existe.

Sorties :

XId-SerV : l'identifiant d'un service.

touche : La dernière touche enfoncée.

Postconditions :

XId-Ser-V : L'identifiant d'un service qui existe.

touche : ∈ { gauche, droite, Esc , Enter}

ALGORITHME

Répéter

Pagination sur les noms de service.
A partir du premier service.

jusqu'à la frappe d'un code de fin cfr touche.

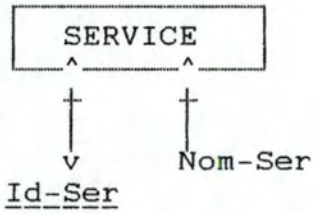
ACCES NECESSAIRES

A l'initialisation :

```
XSerV := SERVICE(:Id-Ser = XId-SerV);
```

Pour la pagination :

```
XSerV := FIRST/LAST/PREV/NEXT SERVICE[
         XId-SerV SORTED (Nom-Ser)];
```



d) **Liste des pages du service visité**

SPECIFICATION : Select-pag

Entrées :

XId-Ser : Identifiant du service courant

XId-SerV : Identifiant du service visité.

Préconditions :

XId-Ser & XId-SerV correspondent à des services existants.

Sorties :

XId-Page : L'identifiant d'une page.

touche : La dernière touche enfoncée.

Postconditions :

XId-Page est une page du service identifiée par XId-Ser-V et est au moins accessible en consultation au service identifié par XId-Ser.

touche ∈ { gauche, droite, esc , return }

ALGORITHME

Répéter

Pagination sur les pages du service identifié par XId-SerV. A partir de la première page accessible, au moins en consultation par le service identifié par XId-Ser.

Chaque ligne est composée du nom d'une page et de l'intervalle de consultation, ainsi que de l'intervalle de modification.

jusqu'à la frappe d'un code de fin cfr RETOUR.

ACCES NECESSAIRES

A l'initialisation :

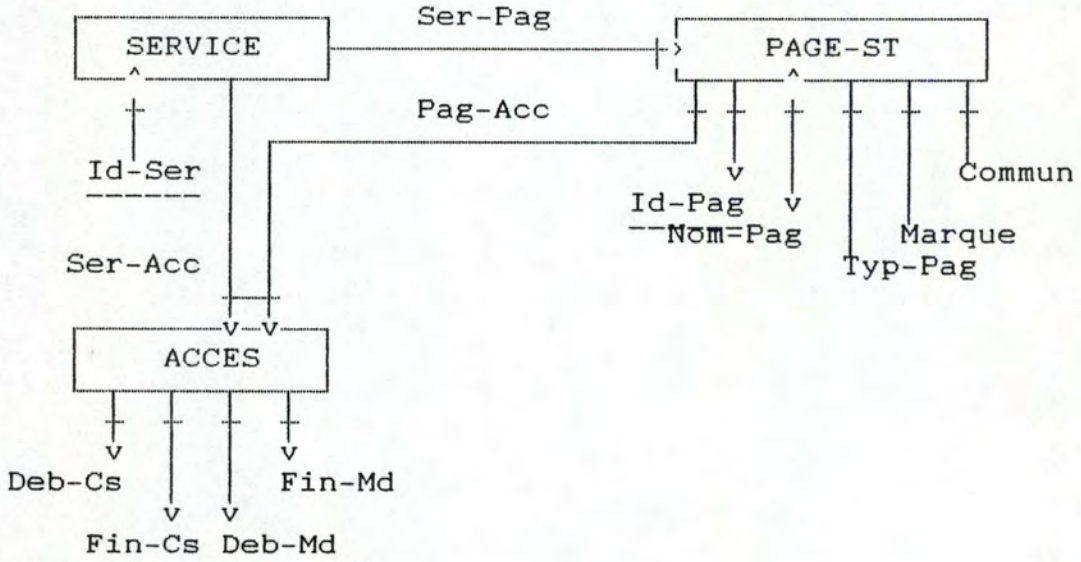
```
XPage := PAGE(SER-PAGE : SERVICE (:Id-SER =XId-Ser)
              AND (:Id-Page = XId-Page));
```

Pour la pagination :

```
XPage := FIRST/LAST/PREV/NEXT PAGE [
         (SER-PAGE : SERVICE (:Id-SER =XId-Ser))
          SORTED (Nom-Pag)];
```

Pour les accès permis :

```
Xacc := ACCES (SER-ACC : SERVICE (:Id-SER =XId-Ser)
        AND (PAG-ACC : PAGE-ST (Id-Page = XId-Page))
```



e) **Modification de la date et pagination**

SPECIFICATION : Select-Dat

Entrées :

XId-Ser : Identifiant du service de l'utilisateur.

XId-SerV : Identifiant du service visité.

XId_Pag : Identifiant d'une page du service visité.

XId-Pat :Identifiant d'un patient.

CHOIX-DATE : Une date par défaut.

Préconditions :

Les différents identifiants doivent correspondre à des éléments existants. Xid-pag de XId-SerV doivent correspondre à une page accessible par le service de l'utilisateur.

CHOIX-DATE : DATE-MIN <= CHOIX-DATE <= DATE-MAX.

Sorties :

CHOIX-DATE : La nouvelle valeur déterminée par l'utilisateur.

touche : La dernière touche frappée.

Postconditions :

CHOIX-DATE : DATE-MIN <= CHOIX-DATE <= DATE-MAX.

touche ∈ {gauche, droite, esc, return, haut, bas, home, end}

ALGORITHME

Afficher le contenu de la page courante du dossier
Si ce n'est pas permis donner la première date accessible
S'il n'existe pas de date accessible avertir l'utilisateur

Répéter
Pagination sur les pages du service identifié par XId-Ser_V. A partir de la première page accessible, au moins en consultation par le service identifié par XId-Ser à la date donnée.

ELSE INPUT(XDateMes);
Si la date existe, afficher le contenu de la page à la date de visite = XDateMes.
Sinon demander à l'utilisateur s'il veut créer une nouvelle date de visite.

Si oui -› Créer une nouvelle date de visite et
afficher cette page.
Si non -› Afficher la page à la première date de
visite disponible.}

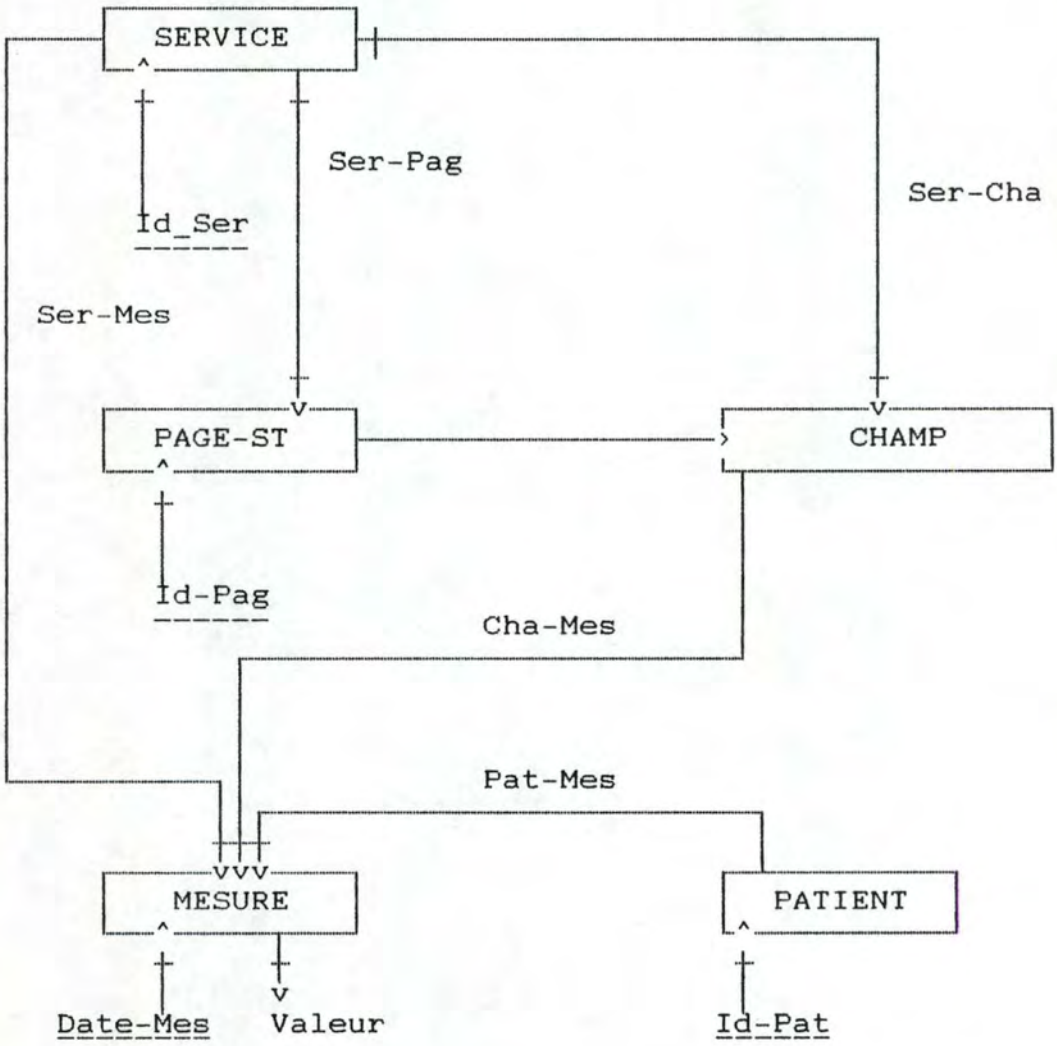
jusqu'à la frappe d'un code de fin cfr RETOUR.

ACCES NECESSAIRES

Pagination sur base des dates :

```
CHOIX-DATE := FIRST/LAST/PREV/NEXT Date-Mes(:Mesure [
                  (PAT-MES : PATIENT(:Id-Pat = XId-Pat)
              AND (SER-MES : SERVICE(:Id-Ser = XId-SerV)
              AND (CHA-MES : CHAMP(: Id-Cha2 = XId-Cha))
              SORTED(Date-Mes)])
```

En fait pour trouver le champ XId-Cha, il est nécessaire encore de passer par la page choisie. D'où le graphe qui suit.



I-3) Actions sur une page

a) **Introduction**

Nous ne détaillerons que les fonctions de modification et de consultation, en effet dans les autres fonctions, si les actions sont différentes elles n'utilisent pas de nouveaux chemins.

b) **Consultation, Modification**

SPECIFICATION : Consultation-Modif-Page

Entrées :

XId-Pag : Identifiant de la page courante.

XDEBCS, XFINCS, XDEBMD, XFINMD : le type d'accès permis sur cette page, ainsi que l'intervalle de temps.

XId-Pat : identifiant du patient.

XId-Ser-V : Identifiant du service visité.

Date-Choix : Date à laquelle s'effectue les actions.

Précondition :

Les différents identifiants fournis en entrée doivent correspondre à des éléments de la base de données.

TYPEACCES : est soit - Cs / Mod / Tout.

Sortie :

Modification possible de la base de données.

Postcondition :

La base de données reste cohérente.

ALGORITHME

Répéter :

Pagination sur la page courante.
CS-MOD-SUP-MAJ En fonction des permissions sur les différentes mesures affichées.
Affiche-Pag(Id-SerV,Id-Pag,Id-Pat,Date-choix);

Jusqu'à la frappe de la touche return ou esc
(Il y a demande de confirmation en cas de modification de mesure)

Appel

Affiche-Pag; Nous ne détaillerons pas cette procédure qui a pour fonction d'afficher une page sur l'écran. La développer ne nous apprendrait rien.

ACCES NECESSAIRES

Consultation :

CS le contenu d'une page revient à :

TR-Acc(Id-Ser,Id-Pag) pour vérifier si l'action est permise.

TR-Pag(Id-Ser,Id-Pag) pour trouver la page.

TR-Cha(Id-Ser,Id-Cha) pour trouver les champs composant la page.

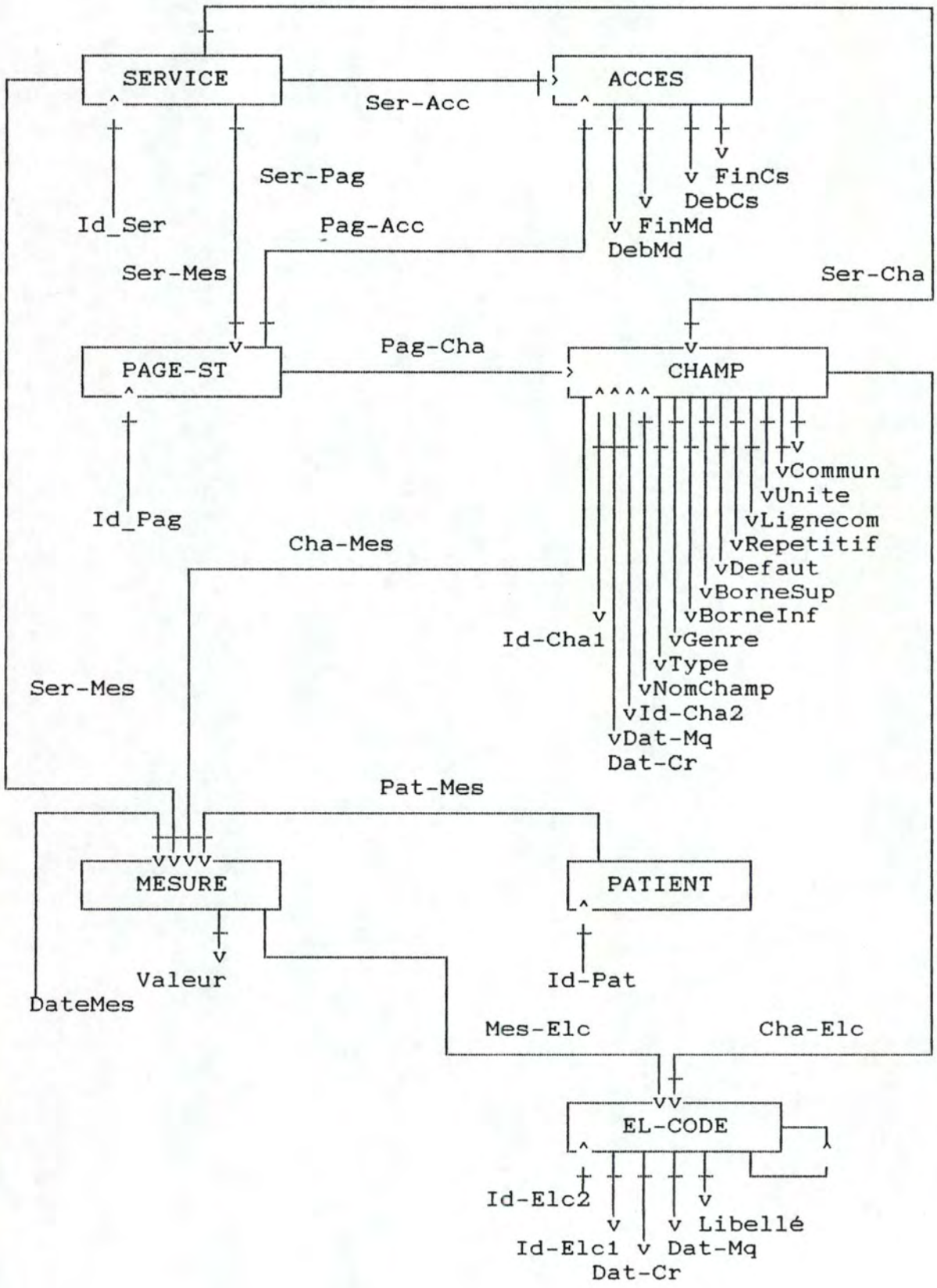
CH-Mes(Id-Ser,Id-Pat,Id-Cha,DateMes) pour rechercher des mesures composant la page.

TR-Elc(Id-Cha,Id-Elc2) si un mesure est de type code.

Modification :

Cr/Md Mesure(Id-Ser,Id-Pat,Id-Cha,DateMes)

On modifie la valeur d'une mesure et si elle n'existe pas on la crée.


SERVICE
ACCES
Ser-Acc
Id_Ser
Ser-Pag
Pag-Acc
FinCs
DebCs
FinMd
DebMd
Ser-Mes
Ser-Cha
PAGE-ST
Pag-Cha
CHAMP
Id_Pag
vCommun
vUnite
vLignecom
vRepetitif
vDefaut
vBorneSup
vBorneInf
vGenre
vType
vNomChamp
vId-Cha2
vDat-Mq
Dat-Cr
Id-Cha1
Cha-Mes
Ser-Mes
Pat-Mes
MESURE
PATIENT
Valeur
DateMes
Id-Pat
Mes-Elc
Cha-Elc
EL-CODE
Id-Elc2
Id-Elc1
Dat-Mq
Dat-Cr
Libellé

c) **Création**

SPECIFICATION : Création-Page

Nous ne détaillerons pas cette fonction, les appels à la base de données étant compris dans ceux de la fonction précédente.

La différence vient du fait que l'on effectue des créations : CR-Mes(Id-Ser,Id-Pat,Id-Cha,DatMes)

d) **Suppression**

ACCES NECESSAIRES : Destruction-Page

Nous ne détaillerons pas cette fonction, les appels à la base de données étant compris dans ceux de la fonction Consultation-Modif-Page.

La différence vient du fait que l'on effectue des destructions : DT-Mes(Id-Ser,Id-Pat,Id-Cha,DatMes)

II) **Structure**

II-1) Le menu

Le menu de cette rubrique permet à l'utilisateur de se diriger vers la fonction de son désir. Développer l'algorithme de cette rubrique ne nous apprendraît rien.

II-2) Pages

a) **Introduction**

SPECIFICATION : Struct-Page

Entrées :

XId-Ser : Identifiant du service de l'utilisateur.

XId-Pag : Identifiant d'une page. (La page par défaut)

Préconditions :

Les identifiants doivent correspondre à des éléments de la base de données.

Sorties :

XId-Pag : L'identifiant de la dernière page utilisée.

Modification possible de la base de données.

Postcondition :

XId-Pag doit correspondre à une page qui existe.

ALGORITHME

```
Répéter
          lecture du clavier;
          SI Touche de Pagination
               ALORS  Afficher la page selon la commande donnée.
          SI Touche Ins ALORS
               choix := ChoixChamp;

               Répéter

                 CASE choix OF

               { Choix d'un champ pour insérer dans la page }
                ChoixChamp : St-Pag-Cha(XId-ser,XId-Cha2,choix);

               { Modification des droits d'accès à la page }
               ChoixAccès : St-Pag-Acc(XId-ser,XId-Pag,choix);

               { Choix de la page courante }
               ChoixPage : St-Pag-Pag(XId-ser,XId-Pag,choix);

          ENDCASE;
        Jusqu'à choix = 'Enter' or 'Esc'
Jusqu'à touche = 'Enter or 'Esc'
```

Appel

-St-Pag-Cha;
-St-Pag-Acc;
-St-Pag-Pag;

ACCES NECESSAIRES

Tous les accès à la base de données sont décentralisés dans les fonctions appelées.

b) **Liste de pages**

SPECIFICATION : ST-Pag-Pag

Entrées :

XId-Ser : Le service qui est propriétaire de cette page.
XId-Pag : Identifiant de la page par défaut.

Précondition :

XId-Ser doit correspondre à un service qui existe.
XId-Pag doit correspondre à une page accessible par le service.

Sortie :

Choix : Correspond à la dernière touche enfoncée par l'utilisateur.

Postcondition :

La base de données peut avoir été changée en cas de modification (CR,SUP,Renommage, modification de statut)

Choix ∈ { Enter, Esc, ChoixAccès , ChoixChamp }

ALGORITHME

Répéter:

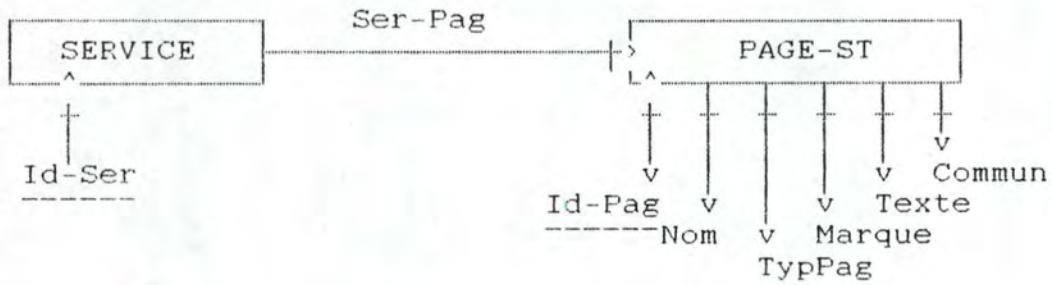
Pagination sur l'ensemble des pages du service ID-SERV pour.
- Consulter.
- Modifier le nom ou statut d'une page (effets rétroactifs).
- Supprimer une page : Dans ce cas l'utilisateur peut décider que cette destruction a ou non des effets rétroactifs.
- Créer une nouvelle page.

Jusqu'à la frappe de la touche 'Enter' -› demande de validation ou de la touche 'esc' = abandon.

ACCES NECESSAIRES

Pagination sur les pages :

```
XPage := FIRST/LAST/PREV/NEXT PAGE [
            (SER-PAGE : SERVICE (:Id-SER =XId-Ser))
             SORTED (Nom-Pag)];
```



c) Les champs dans une page

SPECIFICATION

Entrée :

ID-SERV : Le nom du service qui développe cette page.

ID-PAGE : identifiant de la page que l'on désire modifier dans sa structure.

Préconditions :

ID-SERV & ID-PAGE doivent exister.

Sortie :

ID-PAGE : L'identifiant d'une page du service courant.

Postcondition :

La base de données se voit modifier.

ALGORITHME

Répéter

Positionnement sur une ligne de la page ->
- Soit choix d'un champ et insertion de ce dernier.(Un champ peut se retrouver deux fois dans une page)
- Soit insertion d'un texte de légende.

Jusqu'à la frappe de la touche 'Enter' : en cas de modification demande de validation, ou de la touche 'ESC' qui correspond à un abandon des modifications.

ACCES NECESSAIRES

Pagination sur les champs :

```
XChamp := FIRST/LAST/PREV/NEXT CHAMP [
                (Ser-Cha : SERVICE (:Id-SER =XId-Ser))
                SORTED (Nom-Cha)];
```

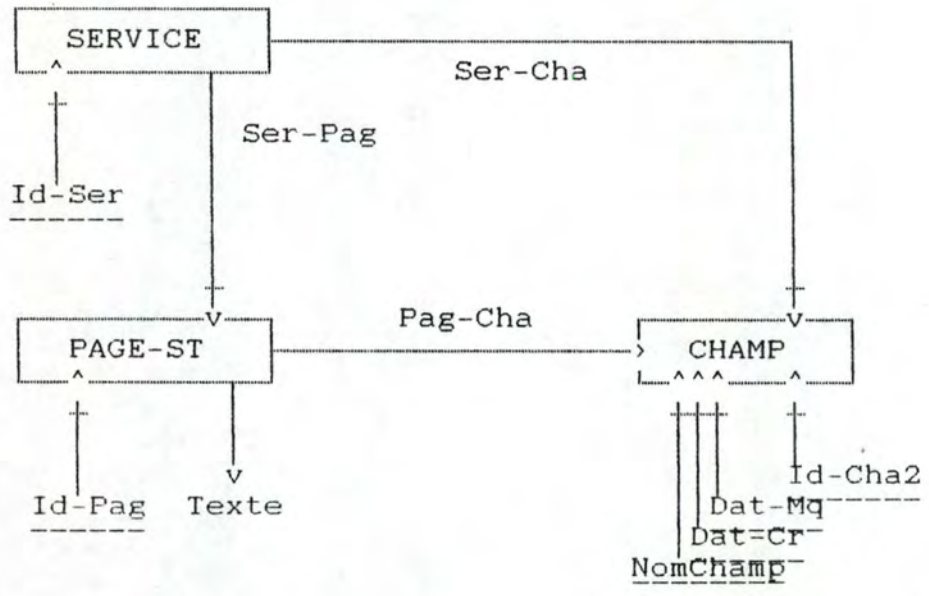
Modification d'une page :

pour du texte :

```
XPage := PAGE [ (Ser-Pag : SERVICE (Id-Ser = XId-Ser)
            AND (PAGE-ST(: TEXTE = XTEXTE)]
```

pour un champ :

```
XPage := PAGE [ (Ser-Pag : SERVICE (Id-Ser = XId-Ser)
            AND (Pag-Cha:CHAMP(Id-Cha2 = XId-Cha2)]
```



d) **Accès**

SPECIFICATION : Accès-Pag

Entrées :

ID-SERV : Le service propriétaire de cette page.

ID-PAGE : Identifiant de la page.

Préconditions :

ID-SERV et ID-PAGE doivent correspondre à un service et à une page respectivement, de la base de données.

Sortie :

Modification, création d'articles de type 'ACCES'.

Postcondition :

La base de données peut avoir été modifiée.

ALGORITHME

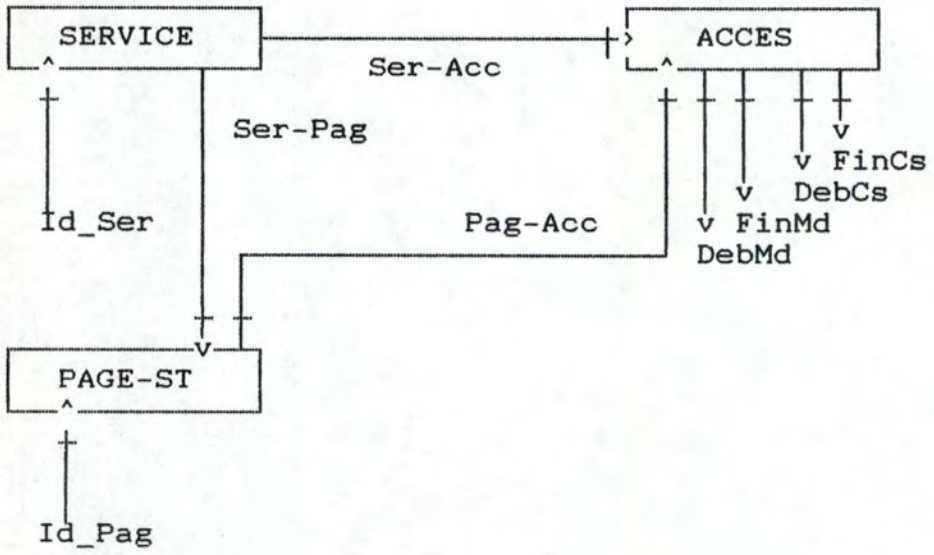
Répéter

Pagination sur les accès, et modification des types d'accès permis sur un intervalle de temps.

Jusqu'à la frappe de la touche 'Enter' -› demande de confirmation des modifications effectuées ou de la touche 'ESC' qui correspond à un abandon des modifications.

ACCES NECESSAIRES

CS/CR/MOD/SUP-ACCES(XId-SerV,XId-Pag)



II-3) Champs

a) Introduction

SPECIFICATION : Str-Cha

Entrées :

ID-SERV : Le service propriétaire du champ.

ID-CHAMP : L'identifiant d'un champ du service ID-SERV.

Préconditions :

Le service et le champ doivent exister.

Sortie :

Des articles de type 'CHAMP' peuvent avoir été mofifiés.

Postcondition :

La base de données peut avoir été modifiée.

ALGORITHME

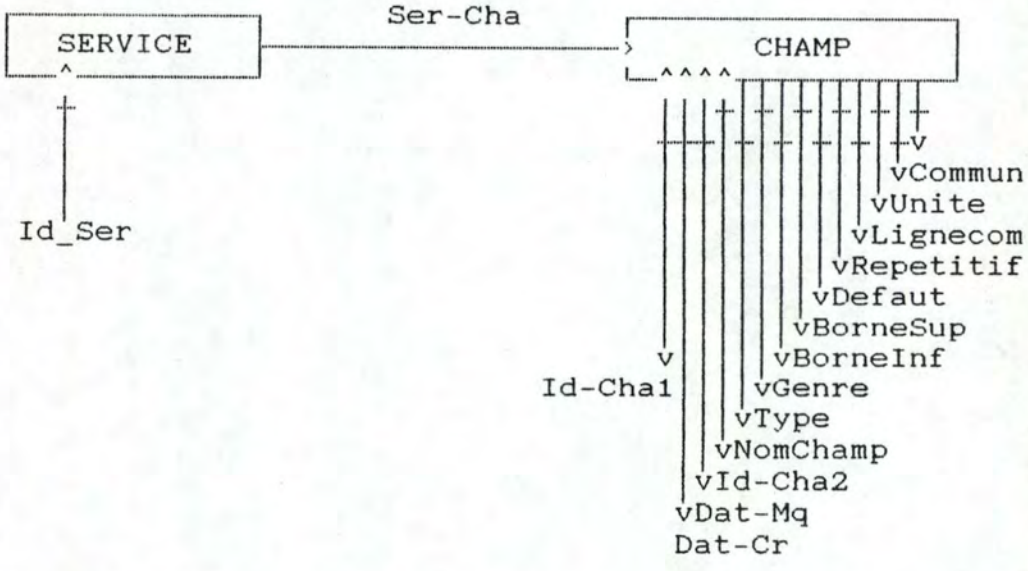
Répéter

Modification de la définition d'un champ.
Ou choix d'un autre champ ('Ins')

Jusqu'à la frappe de :
Enter : Si modification
- Si modif incompatible -› proposition d'une nouvelle version.
- Si reste compatible -› effets rétroactifs
Esc : Abandon.

ACCES NECESSAIRES

CS/CR/MOD/SUP-CHAMP(XId-SerV,XId-Cha)



Nous avons repris dans cet accès tous les accès possibles dans cette rubrique. Nous ne ferons que citer les fonctions de définition des champs.

b) **Champ : 'Nombre'**

c) **Champ : 'Suite de caractères'**

d) **Champ : 'Date'**

e) **Champ : 'Heure'**

f) Liste des champs

SPECIFICATION : Str-Cha-Liste

Entrée :

XId-Ser : Le service qui est propriétaire des pages.

Précondition :

XId-Ser doit correspondre à un service qui existe.

Sortie :

XId-Cha : L'identifiant d'un champ.

Des articles de type 'CHAMP' peuvent avoir été modifiés, créés ou supprimés...

Postcondition :

La base de données peut avoir été changée en cas de modification (CR,SUP,Renommage), mais reste cohérente.

ALGORITHME:

Répéter

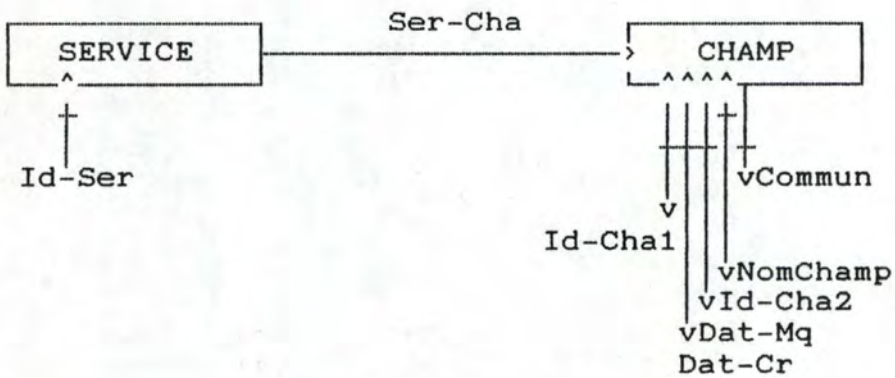
Pagination sur l'ensemble des champs du service XId-Ser pour.

- Consulter.
- Modifier le nom d'un champ(effet rétroactif).
- Supprimer un champ : dans ce cas l'utilisateur peut décider que cette destruction a ou non des effets rétroactifs.
- Créer un nouveau champ.

Jusqu'à la frappe de la touche retour -› demande de validation ou de la touche 'esc' = abandon.

ACCES NECESSAIRES

CS CHAMP(XId-Ser,XId-Cha2)



II-4) Code

a) **Introduction**

OBJECTIF

C'est dans cette fonction que l'utilisateur va créer, modifier ou supprimer un code. L'utilisateur peut aussi compulser le contenu d'un code qui appartient au service commun et dont il n'est pas le propriétaire.

b) **Liste des codes**

SPECIFICATION : Struct-Liste-Code

Entrées :

XId_Ser : Identifiant du service.
XId-Cha : Identifiant d'un code par défaut.

Préconditions :

XId-Ser, XId-Cha doivent correspondre à des éléments de la base de données.

Sortie :

XId-Cha : L'identifiant d'un code sélectionné par l'utilisateur.

Postcondition :

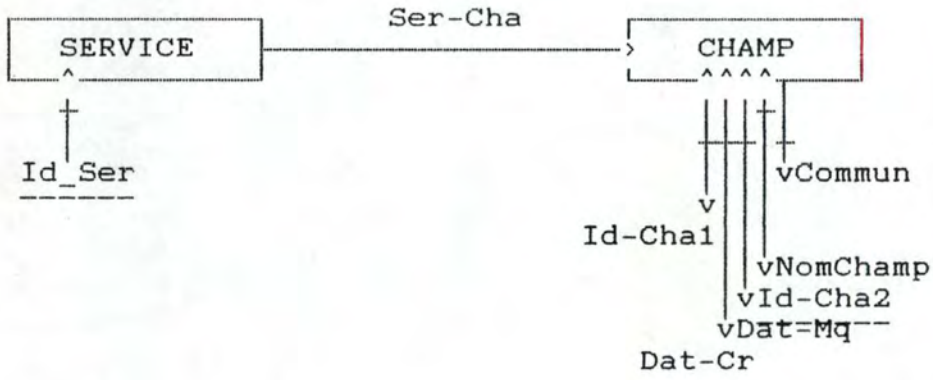
XI-Cha correspond à l'identifiant d'un code de la base de données accessible au service utilisateur.

ALGORITHME

```
Répéter
        Pagination sur les champs accessibles
                    (du service ou commun);
Jusqu'à la frappe d' 'Enter' ou 'Esc'
```

ACCES NECESSAIRES

CS-CODE(XId-Ser,XId-Cha2)



c) **Modification d'un code**

SPECIFICATION : Struct-Code

Entrées :

XId_Ser : Identifiant du service.
XId-Cha : Identifiant d'un code par défaut.

Préconditions :

XId-Ser, XId-Cha doivent correspondre à des éléments de la base de données.

Sortie :

XId-Cha : L'identifiant d'un code sélectionné par l'utilisateur.

Postcondition :

XI-Cha correspond à l'identifiant d'un code de la base de données accessible au service utilisateur.

ALGORITHME

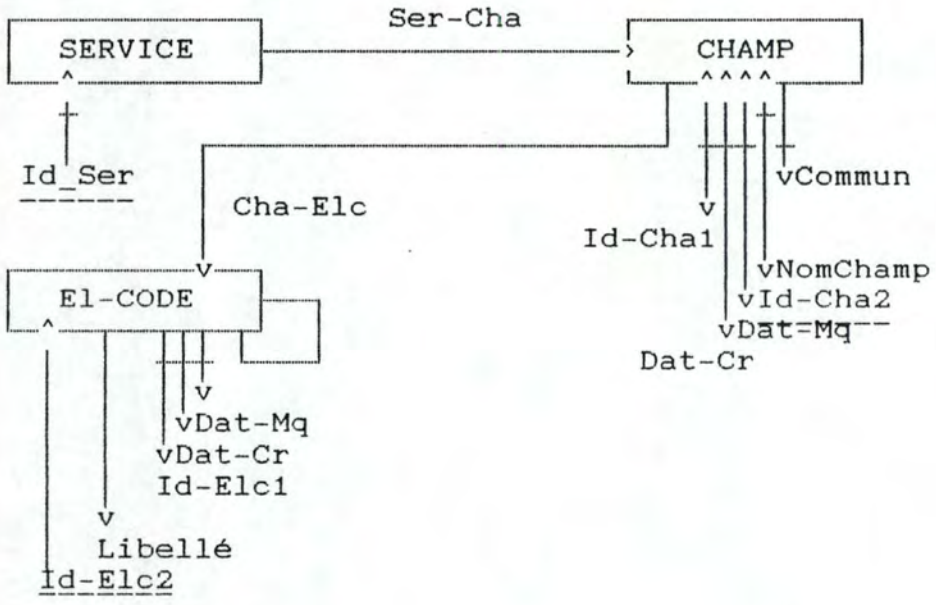
Répéter

Modification de la constitution du code

Jusqu'à la frappe de 'Enter' ou 'Esc'

ACCES NECESSAIRES

CS/CR/DT/MD-El-Code(XId-Ser,XId-Cha2,XId-Elc2)



III) **Impression**

III-1) Le menu

Pour cette rubrique nous ne détaillerons pas les algorithmes pour trouver les accès nécessaires à la base de données. En effet, les accès propres à cette rubrique sont déjà repris dans la consultation, à l'exception de la modification de la longueur d'une page. Donc nous ne citerons que les fonctions.

III-2) Les fonctions

a) Longueur d'une page

SPECIFICATION :

Entrée :

XId-Ser : Identifiant d'un service.

Précondition :

XId-Ser doit correspondre à l'identifiant d'un service.

Sortie :

Modification d'un article Service.

Postcondition : /

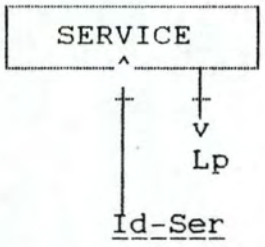
ALGORITHME

Répéter

Lecture de la nouvelle longueur de page.

Jusqu'à la frappe de 'Enter' ou 'Esc'

ACCES NECESSAIRES



b) **Liste des patients**

c) **Liste des services**

d) **Liste des pages**

e) **Liste des champs**

IV) **Divers**

IV-1) Le menu

Pour cette rubrique nous ne détaillerons pas les algorithmes pour trouver les accès nécessaires à la base de données. En effet, les accès propres à cette rubrique sont déjà compris dans les rubriques détaillées précédemment à l'exception des cas de modification, mais ces derniers cas n'ajoutent pas de nouveaux accès nécessaires. Nous nous contenterons de citer les fonctions.

IV-2) Les fonctions

a) **Archivage**

b) **Reprise**

c) **Modification du mot de passe**

SPECIFICATION : Modif-Mot-De-Passe

Entrée :

XId-Ser : Identifiant d'un service.

Précondition :

XId-Ser doit correspondre à l'identifiant d'un service.

Sortie :

Modification d'un article Service.

Postcondition : /

ALGORITHME

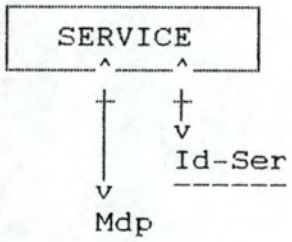
Répéter

Lecture du nouveau mot de passe
+ Vérification de la connaissance de l'ancien mot de passe.

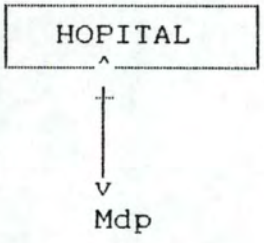
Jusqu'à la frappe de 'Enter' ou 'Esc'

ACCES NECESSAIRES

Pour un service :



Pour l'Hôpital :



d) **Modification de la liste des patients**

e) **Modification de la liste des services**

V) **Sortie**

Cette rubrique n'apporte rien au niveau des accès nécessaires à la base de données.

2.4 Journal

Le journal ne sera pas développé, en effet, le développement d'un tel outil est lié au logiciel de gestionnaire de base de données.

Modèle des Accès Nécessaires: *HOPIDOS*

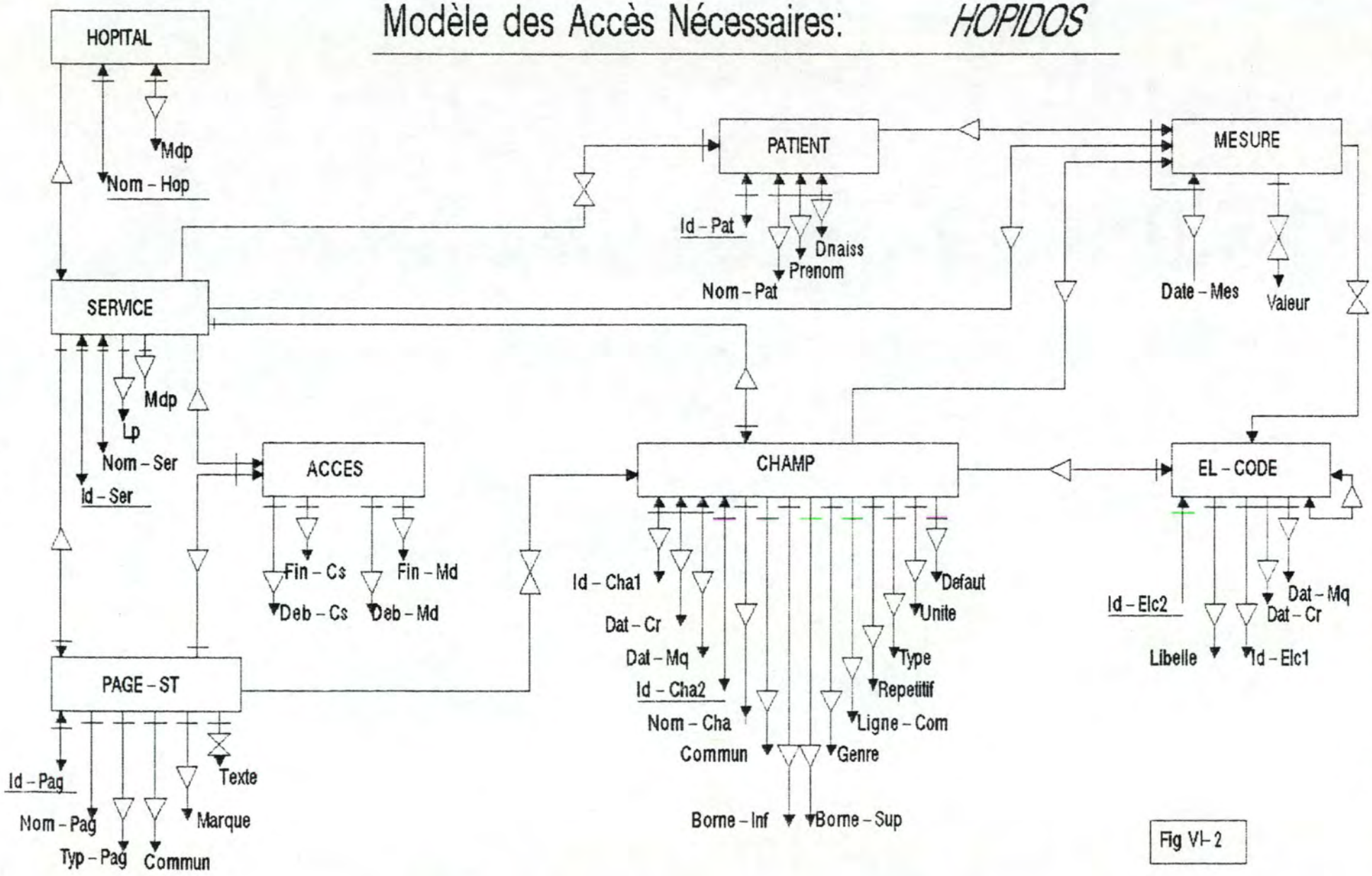


Fig VI-2

CONCLUSIONS

1 Ce que m'a apporté le mémoire

Un mémoire est une oeuvre toujours très intéressante pour un étudiant. Il est enfin confronté à un problème de taille réelle où rien n'est laissé sous silence.

J'ai pu observer un centre informatique de près et voir les problèmes quotidiens auxquels il est confronté.

J'ai vu des vrais utilisateurs, j'ai dialogué avec eux. J'ai rencontré au sein de l'hôpital des médecins très conscients de la puissance de l'outil informatique. Chaque discussion avec ces derniers était pour moi des plus enrichissantes.

Et enfin, j'ai dû faire face aux difficultés inhérentes à la réalisation d'un logiciel qui devait correspondre aux désirs des utilisateurs.

2 Améliorations et extensions

Supposons que le logiciel ait pu être réalisé grâce à des outils fiables, efficaces et conviviaux.

Il serait alors intéressant de lui ajouter divers outils tels que :

- Un traitement de texte permettant à un médecin de créer une lettre type qui se remplirait automatiquement des mesures effectuées. Ceci permettrait d'écrire des formulaires pré-établis pour, par exemple, les médecins traitant.
- Un agenda pour les consultations.
- Une gestion de stocks de médicaments avec mise à jour automatique.
- Permettre l'élaboration de base de données statistiques.
- ...

3 Bilan du mémoire

Nous n'avons qu'un regret : ne pas avoir pu disposer d'outils appropriés pour faire aboutir le mémoire à la réalisation du logiciel, même réduit.

Nous disposions d'outils d'une 'ancienne' génération pour créer des logiciels d'une 'nouvelle' génération, celle des produits à 'géométrie variable'. Enfin l'homme ne doit plus se plier à une informatique lourde mais c'est elle qui doit être la plus conviviale possible.

Mais, nous sommes trés contents de l'expériene vécue, nous nous sommes sentis passionnés jusqu'au bout par ce travail. Nous avons rencontré un ensemble de personnes enthousiastes aussi bien chez les médecins que chez les informaticiens.

Pour ce qui est de l'étude du problème et de son analyse, le mémoire a satisfait nos espoirs, mais pour ce qui est de la réalisation nous n'avons pas trouvé le matériel idéal ni le langage de programmation adéquat.

Mais si le mémoire ne débouche pas sur une réalisation d'un logiciel, nous espérons que le travail accompli ne restera pas sans lendemain et qu'un jour il y aura moyen de l'implémenter ou que l'étude du problème faite dans ce mémoire pourra servir de matière à réflexion pour un autre logiciel.

Bibliographie.

[BODART-PIGNEUR] : F. Bodart & Y.Pigneur : 'Conception assistée des applications informatiques: 1. Etude d'opportunité et analyse conceptuelle'. Masson ; Presses Universitaires de Namur. 1983.

[HAINAUT] : J.-L. Hainaut : 'Conception assistée des applications informatiques: 2. Conception de la base de données'. Masson ; Presses Universitaires de Namur. 1986.

[ROGER] : Francis Roger : 'Le résumé du dossier médical - Indicateur informatisé de performances et de qualité de soins' (Université Catholique de Louvain).

Autres :

'Le résumé Clinique Automatisé'.
Rapport de synthèse des travaux du centre Informatique Médicale de l'Université Catholique de Louvain. Mai 1976.

[DEGRAVE] Rapport Commun rédigé par Dr. E. Degrave : 'Développement interhospitalier du Résumé Clinique Automatisé'. Services du premier ministre ; programmation de la politique scientifique. Rue de la science 8, 1040 BRUXELLES.

FACULTÉS
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

Annexe :

Gestion Informatisée de Dossiers Médicaux au sein d'un hôpital

Jean-François GOGUIN

Annexe au mémoire présenté en vue de l'obtention du diplôme de Licence en Informatique.

Promoteur : Prof. J. FICHEFET

Chef de projet : Mr. Claude HENRY

ANNEE : 1986-1987

Adresse : rue Grandgagnage, 21, B-5000 NAMUR (Belgium) - ☎ 081 - 22 90 65 - Télex 59.222 FAC NAM-B

PLAN DE L'ANNEXE

1 Introduction générale
- 1.1 Introduction : Idées et options
- 1.2 Architecture
- 1.3 Conclusions

2 Le programme
- 2.1 Définitions
- 2.2 Niveau 7
  - 2.2.1 Contenu
  - 2.2.2 Procédures et Spécifications
- 2.3 Niveau 6
  - 2.3.1 Contenu
  - 2.3.2 Procédures et Spécifications
- 2.4 Niveau 5
  - 2.4.1 Contenu
  - 2.4.2 Procédures et Spécifications
- 2.5 Niveau 4
  - 2.5.1 Contenu
  - 2.5.2 Procédures et Spécifications
- 2.6 Niveau 3
  - 2.6.1 Contenu
  - 2.6.2 Procédures et Spécifications
- 2.7 Niveau 2
  - 2.7.1 Contenu
  - 2.7.2 Procédures et Spécifications
- 2.8 Niveau 1
  - 2.8.1 Contenu
  - 2.8.2 Procédures et Spécifications

3 Les Fichiers annexes
- 3.1 Les Menus
- 3.2 Les Ecrans d'Aide et de Démonstration

1 Introduction : Idées et options

La maquette a été construite dans plusieurs optiques :

- Concevoir un logiciel convivial, pour ce faire j'ai adopté une approche essai-erreur. Le logiciel fut créé à partir d'idées déterminées, j'avais bien défini les fonctions qui s'y trouveraient, mais pas la forme sous laquelle elles se présenteraient à l'utilisateur. La première étape fut la constitution de petites procédures-outils me permettant de façonner mon programme selon mes désirs.

- La seconde optique était de créer un programme fiable et utilisant un matériel le plus répandu sur le marché, j'ai nommé les PCs, avec un compatibilité au logiciel le plus standard en matière de gestion de base de données : DBASE-III. De ce fait si des options statistiques, graphiques manquaient, un utilisateur pouvait utiliser sa base de données avec un logiciel à outils intégrés.

Pour ces deux raisons, j'ai choisi de reprendre le design d'un logiciel à outils intégrés : FRAMERWORK. Ainsi pour ne pas dépayser l'utilisateur, j'ai décidé de prendre des commandes et des écrans assez proches de ce logiciel.

Il me fallait pour ce faire, un langage me permettant de travailler à un niveau assez bas : TURBO-PASCAL. Ce dernier dispose d'outils permettant de gérer des fenêtres, mais nécessite alors une carte graphique qui n'existe pas sur la version de base des PCs. J'ai donc du recréer des outils de gestion de fenêtres.

Si ce langage m'a permis de créer mes écrans selon mes désirs, il a aussi l'inconvénient qui va de paire avec cet avantage : une longueur de programme, plus de 3000 lignes uniquement pour la gestion des écrans. Il faut dire que ces lignes ne sont que du programme, car les écrans d'aide et les menus sont contenus dans des fichiers annexes au programme, cette option fut prise afin de rendre les modifications plus aisées.

Cette annexe se compose comme suit :

Dans les trois premiers points, je présente la maquette.

Suite à ceci, je présente le logiciel documenté, en commençant par la partie définition des variables, en continuant avec les différents niveaux et je termine avec les écrans d'aide et les écrans de démonstration.

2 Architecture

Cette maquette est décomposée en plusieurs niveaux. Chaque procédure appartenant à un niveau n'utilise que des procédures de niveau inférieur et son fonctionnement est indépendant des procédures qui peuvent l'utiliser.

NIVEAU 1 :

RACINE

FONCTIONS : Consultation / Structure / Impression / Autres /Sortie

NIVEAU 2 :

Les procédures propres à chaque fonction.

NIVEAU 3 :

Procédures de gestion des menus (Lecture et utilisation)

NIVEAU 4 :

Procédures de lecture particulières (Date, Heure...)

NIVEAU 5 :

Aide, gestion des erreurs et des fichiers texte

NIVEAU 6 :

Gestion des fenêtres.

NIVEAU 7 :

Lecture du clavier.

3 Conclusions

La maquette a rempli patiellement mes désirs ; je n'ai pas un programme maquette gérant une base de données, mais il a montré qu'il était possible de construire un logiciel de gestion de dossiers médicaux trés simple d'utilisation, une heure suffit pour apprendre les diffférentes commandes. De plus cette simplicité n'empêche pas un grand nombre de possibilités.

Si la conception des idées clefs du logiciel se sont faites indépendamment de toute implémentation, la réalisation a fortement souffert des contraintes des logiciels et de la machine. Il eut fallu un logiciel de quatrième génération, mais ceux-ci sont rares et bien souvent n'ont pas atteint un stade de maturité suffisant (lourdeur, peu fiable, possibilités trop limitées). Mais je reste très optimiste, les prochains langages devraient combler tous ces problèmes et nous permettre de réaliser des logiciels facilemment, rapidemment, tout en étant de plus en plus fiables.

ontenu

Cette partie contient les constantes, les types de variables définis et les variables globales.

a partie définition des constantes, types et variables

```
*------------------------------------------------------------------------*)
*                                                                        *)
*          ***********          CONST          ***********              *)
*                                                                        *)
*------------------------------------------------------------------------*)
onst
* adresses physiques *)
    adec =$B800; (* adresse physique du premier caractère de l'écran *)
    tecran = 2000 ; (* 25*80 taille de l'écran en caractères *)

* Les codes pour l'affichage en inverse et en normal *)
    inv  = $70; (* $71 ou $70 fonction du type de PC *)
    vrai = $F;  (* $E ou $F   fonction du type de PC*)

* Pour les menus *)
* les types *)
    Hz  = 1; (* les genres d'écrans possibles *)
    V   = 2;
    HzV = 3;
* les numéros des menus *)
    nmenus = 25 ;     (* nombre de menus*)

    me_prem = 1     ; (* menu principal *)
    me_cons = 7     ; (* menu consultation pat-serv-page-date *)
    me_cs_pat = 8   ; (* menu liste de patients  *)
    me_cs_ser = 9   ; (* menu liste de services  *)
    me_cs_pag = 10  ; (* menu liste de pages     *)
    me_cs_dat = 11  ; (* menu date ou intervalle *)

    me_s_pg   = 12 ; (* menu structure - page *)
    me_s_lpg  = 10 ; (* menu structure - liste de pages *)
    me_s_lch  = 13 ; (* menu structure - liste de champs *)
    me_s_lser = 14 ; (* menu structure - liste de services *)

    mes_ch_en = 16;  (* menu structure d'un champ 'entier'*)
    mes_ch_re = 16;  (* menu structure d'un champ 'réel'  *)
    mes_ch_ch = 17;  (* menu structure d'un champ 'chaîne'*)
    mes_ch_da = 18;  (* menu structure d'un champ 'date'  *)
    mes_ch_he = 19;  (* menu structure d'un champ 'heure'  *)
    mes_ch_co = 20;  (* menu structure d'un champ 'code'  *)

    mesc_tip   = 21 ; (* menu structure lecture du champ 'tip' *)
    mesc_genre = 22 ; (* menu structure lecture du champ 'genre' *)
    mesc_on    = 23 ; (* lecture d'un choix OUI - NON *)

* dates accepables *)
    preman = 1980;  (* première *)
    dernan = 2100;  (* dernière *)
    mindate = '19800101'; (* date min *)
    maxdate = '21001231'; (* date max *)
    minheure = '000000';  (* heure minimum *)
    maxheure = '245950';  (* heure maximum *)
```

```
* Lecture *)
    maxreel  = 1e37; (* reel max acceptable *)
    longnbre = 14;  (* longueur d'un nombre +.........E+.. *)
    longmant = 9;   (* longueur de la mantisse sans le signe *)
    lnomch   = 10;  (* longueur d'un nom de champ *)
    lchainec = 78 ; (* longueur d'une chaîne de caractères standard *)
    llignecom = 50; (* longueur d'une ligne de commentaire *)
    lunite    = 10; (* longueur d'une unité *)
    lnomdecode = 8 ;(* longueur d'un nom de code *)
*-------------------------------------------------------------------------*)
*                                                                         *)
*          ***********          TYPE          ***********                *)
*                                                                         *)
*-------------------------------------------------------------------------*)
ype
* type d'utilisation générale *)
* chaînes de caractères *)
    ligne      = string[90];
    nomfichier = string[14];
    refaide    = string[3];
    date       = string[8];
    heure      = string[6];
    nomdecode  = string[lnomdecode];

* type pour la lecture des données *)
  TOUCHECTRL = (F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,
                CTRA,CTRB,CTRC,CTRD,CTRE,CTRF,CTRG,CTRH,CTRI,CTRJ,
                CTRK,CTRL,CTRM,CTRN,CTRO,CTRP,CTRQ,CTRR,CTRS,CTRT,
                CTRU,CTRV,CTRW,CTRX,CTRY,CTRZ,ESC,HOME,FIN,HAUT,BAS,
             GAUCHE,DROITE,PGUP,PGDN,INS,DEL,DELETE,RETURN,CTGAUCHE,CTDROITE,
                CTPGUP,CTPGDN,autre);
  SOFCTRL = SET OF TOUCHECTRL;
  SOFC    = SET OF CHAR;

* type divers *)
    tampon = ^char;
    point  = record
                l,c : integer;
             end;

* type pour l'utilisation de la pile des fenêtres *)
  ptrelemstack = ^elemstack;
  elemstack    =record
                   pr : ptrelemstack;
                   w  : tampon;
                   hg,bd : point;
                 end;
  stack = ptrelemstack;

* type pour les menus *)
    naide = nomfichier;
    pch   = ^choix;
    choix = record prec,suiv : pch;
                   de,a      : integer;
                   comm      : tampon; (* commentaire associé à l'écran *)
                   num       : integer; (* numéro du choix *)
                   mesuiv    : integer; (* désigne le numéro du menu
                                           associé à ce libellé en cas d'écran HzV *)
                   naide     : refaide; (* référence pour l'écran d'aide *)
```

```
            end;
(*
 structure :
   choix.comm : Le premier byte pointé est la longueur du commentaire,
                les bytes suivants étant le commentaire
*)
      menu = record
                  hg,bd : point;
                  genre : integer; (* le genre d'écran : Hz, V ou HzV. *)
                  ch    : pch;     (* Pointe vers le premier choix *)
                  pme   : tampon;  (* pointe vers l'écran *)
                  dc    : pch;     (* dernier choix dans cet écran *)
                  nme   : integer; (* numéro du menu  *)
                  lcomm : integer; (* position de la ligne de commentaire
                                      0 s'il n'en n'existe pas *)
                  naide : refaide; (* reférence pour l'écran d'aide *)
               end;
    tmenus = array[1..nmenus] of menu;

(* lecture *)
    chainec = string[lchainec];
    nbrec   = string[longnbre];
    lignecomment  = string[llignecom];

(* l'utilisation des champs *)
    typ = (entier,reel,chaine,dat,heur,code);
    genr = (statique,dynamique);
    champ = record
              nom : string[8];
              genre : genr;
              repetitif : boolean;
              commun    : boolean;
              lignecom  : lignecomment;
              tip       : typ;
              unite     : string[lunite];

 * entier *)  intinf,intsup,intdef : integer;

 * réel *)    reelinf,reelsup,reeldef : real;

 * chaîne *)  chdef : chainec;

 * date *)     dateinf,datesup,datedef : date;

 * heure *)   heureinf,heuresup,heuredef : heure;

 * code *)    nomcode  : nomdecode;
              quantite : boolean;
              codedef  : integer;
            end;

    version = (floppy,hard);

*------------------------------------------------------------------------*)
*                                                                        *)
*          ***********          VAR          ***********                *)
*                                                                        *)
*------------------------------------------------------------------------*)
ar
   st : stack ;      (* pointe sur le dernier élément de la pile *)
```

```
   final : boolean; (* indique la terminaison du programme *)
   tp : tmenus;     (* tableau contenant les écrans du menu principal *)
   buffer : tampon; (* buffer où l'on met les écrans temporellement *)
   f : text;        (* fichier texte à usages multiples *)
* date *)
   datejour : date ;(* date du jour *)
   premdate,derndate : date;

* debug *)
   vers : version; (* permet de travailler sur disque dur
                      ou sur deux disquettes *)
   ecran_ext    : string[3]; (* extension du fichier des écrans *)
   tc : touchectrl;
```

ontenu

Nous trouvons à ce niveau les procédures du plus bas niveau pour la lecture du clavier...

Notation :
-Touche Normale : L'ensemble des caractères du code ASCII compris entre 32 et 126.

- Touche de contrôle : elles sont définies dans le paragraphe décrivant DEF.PAS.

Programme et spécifications

```
(*-------------------------------------------------------------------------*)
(*                                                                         *)
(*          ***********      NIVEAU   70              ***********         *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)

(*  >>>>>>>>>>>>>>>>>>>>>>>>   CLAVIER       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<  *)
(*  ┌-----------------------------------------------------------------┐    *)
(*  ¦ NIVEAU 72                CLAVIER                               ¦    *)
(*  └-----------------------------------------------------------------┘    *)
PROCEDURE CONTROLE(var T:TOUCHECTRL ; var C:CHAR);
(*   OBJECTIF  :  Lecture  au  clavier  d'une  touche  de  type normal
(alphanumérique)  ou  de  type  contrôle  (touches  contrôles...), ces
touches doivent appartenir au type 'TOUCHECTRL' cfr DEF.PAS.

     Entrées          : /

     Sorties          : T : La touche de contrôle qui vient d'être lue.
                        C : La touche normale qui vient  d'être lue.

     Postconditions :  Si la touche qui  vient d'être lue  est de type
normal, T contient la valeur 'autre'.
*)
VAR CH2 :CHAR;

BEGIN
    reset(KBD);
    read(KBD,C);
    if keypressed (* une touche à 2 codes ? *)
    THEN BEGIN
            read (KBD,CH2);
            CASE ORD(CH2) OF
         59..68 : T := TOUCHECTRL(ORD(CH2)-59);
             71 : T := HOME;
             72 : T := HAUT;
             73 : T := PGUP;
             75 : T := GAUCHE;
             77 : T := DROITE;
             79 : T := FIN;
             80 : T := BAS;
             81 : T := PGDN;
             82 : T := INS;
             83 : T := DEL;
            115 : T := CTGAUCHE;
            116 : T := CTDROITE;
```

```
          118 : T := CTPGDN;
          132 : T := CTPGUP;
           ELSE T := autre;
       END
    END
    ELSE
       CASE ORD(C) OF
       1..7 : T := TOUCHECTRL(ORD(C)+9);
          8 : T := DELETE;
      9..12 : T := TOUCHECTRL(ORD(C)+9);
         13 : T := RETURN;
     14..26 : T := TOUCHECTRL(ORD(C)+9); (* CRTL U NE FONCTIONNE PAS *)
         27 : T := ESC;
         ELSE T := autre;
       END;
END;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 70                                                            I *)
(* I                          CLAVIER                                     I *)
(* I                                                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure cachecurs;
(*   OBJECTIF : Faire disparaître le curseur de l'écran.

     Postconditions :  Le caractère en position (79,25) est de couleur
noir sur fond noir.
*)
begin
  textcolor(0);
  gotoxy(79,25);write(' '); gotoxy(79,25);
  textcolor(15);
end;

procedure wait;
(*   OBJECTIF :  Attente de la frappe d'une touche quelconque, pendant
cette attente le curseur reste invisible.

     Postconditions :  A la sortie de  cette procédure le  curseur est
caché.
*)
var c : char;
begin
    cachecurs;
    while not keypressed do;
    reset(kbd);
    read(kbd,c);
    cachecurs;
end;

PROCEDURE lect(SC: SOFC; SCTRL : SOFCTRL ; var CH:char ; var TC :touchectrl);
 *   OBJECTIF :Procédure de lecture de caractères de 'contrôle'  et de
aractères  'normaux'.   Le  caractère  lu  n'est  accepté  que  s'il
ppartient à SC,  si c'est un caractère normal ou à SCTRL  si c'est un
aractère de contrôle.

     Entrées        : SC   :   ensemble   des   caractères   'normaux'
cceptables.
                      SCTRL :  Ensemble des  caractères  de 'contrôle'
cceptables.
```

```
     Préconditions   : /

     Sorties         : CH : Le caractère normal lu.
                       TC : Le caractère de contrôle lu.

     Postconditions :  Si un caractère  normal est lu,  TC contient la
valeur 'autre'.
*)

BEGIN
    REPEAT
       CONTROLE(TC,CH);
    UNTIL (((CH IN SC) AND (TC = autre)) OR (TC IN SCTRL));
END;

procedure BEEP;
(*   OBJECTIF : Produire pendant un cours instant un BEEP sonore.
*)
begin
   sound(2000);
   delay(80);
   nosound;
end;
```

ontenu

Ce niveau contient toutes les procédures permettant de travailler avec des fenêtres.

REMARQUE :
HG et BD représentent des points décrivant une fenêtre. HG est le point Haut Gauche et BD est le point Bas Droit de la fenêtre.

A chaque utilisation de ces points on présuppose que hg et bd sont compris dans l'écran. Ces variables doivent respecter :

$$0 \leq HG.L \leq BD.L \leq 24$$
$$0 \leq HG.C \leq BD.C \leq 79$$

Les différente procédures utilisant la pile de fenêtres présupposent que cette dernière existe et aucune vérification n'est faite pour savoir s'il reste de la place.

rogramme et spécifications

```
*-------------------------------------------------------------------------*)
*                                                                         *)
*          ***********      NIVEAU   60              ***********         *)
*                                                                         *)
*-------------------------------------------------------------------------*)
*  >>>>>>>>>>>>>>>>>>>>>>  OUTILS WINDOW  <<<<<<<<<<<<<<<<<<<<<<<<<<<<  *)
*  ┌-----------------------------------------------------------------┐    *)
*  ¦ NIVEAU 62                OUTILS WINDOW                        ¦    *)
*  └-----------------------------------------------------------------┘    *)
unction taille(hg,bd : point) : integer;
*   OBJECTIF        :  Fournit la taille en bytes de la fenêtre décrite par hg &
d.  On ne mémorise pas les couleurs...

    Entrées         : HG, BD : Description de la position de la fenêtre.

    Sorties         : La taille de la fenêtre en bytes.

    Postconditions :  La taille réelle de la fenêtre est deux  fois plus grande
ar il existe les couleurs.
)
egin
    taille := (bd.l-hg.l+1) * (bd.c-hg.c+1);
nd;
*  ┌-----------------------------------------------------------------┐    *)
*  ¦ NIVEAU 60                                                       ¦    *)
*  ¦                        OUTILS WINDOW                            ¦    *)
*  ¦                                                                 ¦    *)
*  ¦                                                                 ¦    *)
*  └-----------------------------------------------------------------┘    *)
rocedure pushw(hg,bd : point; var st : ptrelemstack);
*   OBJECTIF        : Met une fenêtre dans la pile st.

    Entrées         : HG, BD : Description de la position de la fenêtre.
                      ST  : La pile sur laquelle il faut mettre la fenêtre.

    Sortie          : ST : La pile mise à jour.
```

```
ar  elst : ptrelemstack;
    l,c,i : integer;
    de,a : ^char;
    long : integer;
    ofsde : integer;

egin
    new(elst);
    elst^.pr := st;
    st := elst;
    elst^.hg := hg;
    elst^.bd := bd;
    getmem(elst^.w,2*taille(hg,bd));
    if taille(hg,bd) <> tecran
    then
      begin
        i := 0;
        long := 2 * (bd.c - hg.c + 1);
        ofsde := 2*hg.c - 162;
        for l := hg.l to bd.l do
        begin
          de := ptr(adec,ofsde+l*160);
          a  := ptr(seg(elst^.w^),ofs(elst^.w^)+i);
          move(de^,a^,long);
          i := i + long;
        end
      end
    else
      begin
       de := ptr(adec,0);
       a := elst^.w;
       move(de^,a^,4000);
      end;
nd;

rocedure popw(var st : stack);
*   OBJECTIF        : Retire la dernière fenêtre mise sur la pile.

    Entrées         : ST : La pile où se trouve la fenêtre.

    Sorties         : ST : La pile mise à jour.

    Postconditions  :  La fenêtre  qui  vient  d'être  retirée  est  affichée à
écran.

ar  elst : ptrelemstack;
    l,c,i : integer;
    de,a : ^char;
    long,ofsde : integer;

egin
    elst := st;
    st := elst^.pr;

    if taille(elst^.hg,elst^.bd) <> tecran
    then
    begin
      i := 0;
        long := 2 * (elst^.bd.c - elst^.hg.c + 1);
        ofsde := 2*elst^.hg.c - 162;
```

```
      for l := elst^.hg.l to elst^.bd.l do
        begin
           a  := ptr(adec,ofsde+l*160);
           de := ptr(seg(elst^.w^),ofs(elst^.w^)+i);
           move(de^,a^,long);
           i := i + long;
        end
      end
    else
      begin
       a := ptr(adec,0);
       de := elst^.w;
       move(de^,a^,4000);
   end;

    freemem(elst^.w,2*taille(elst^.hg,elst^.bd));
    mark(elst);
    release(elst);

nd;

rocedure clsw(hg,bd : point);
*   OBJECTIF        : Effacer sur l'écran une fenêtre.

    Entrées         : HG, BD : Description de la position de la fenêtre.

    Postconditions :  La fenêtre décrite par HG et BD est remplie de caractères
lancs (code ASCII 32).
)
ar l,c : integer;
egin
    for l := hg.l to bd.l do
        for c:= hg.c to bd.c do
           mem[adec : 2*(c+l*80-81)] := 32;
nd;

rocedure cltopst(var st : stack);
*   OBJECTIF        : Efface la dernière fenêtre entrée sur la pile.

    Entrées         : ST : La pile où se trouve la fenêtre.

     Sorties         : ST : la pile mise à jour.

    Postconditions : La dernière fenêtre rentrée n'existe plus.
)
ar  elst : ptrelemstack;

egin
    elst := st;
    st := elst^.pr;

    freemem(elst^.w,taille(elst^.hg,elst^.bd));
    mark(elst);
    release(elst);
nd;

rocedure clst(var st : stack);
*   OBJECTIF        : Vide la pile de toutes ses fenêtres.

    Entrées         : ST : la pile à vider.
```

```
    Sorties         : ST : la pile mise à jour.

    Postconditions : La pile ST ne contient plus de fenêtres.
)
egin
  while st <> nil do cltopst(st);
nd;


rocedure initw(var st: stack);
*   OBJECTIF        : Initialisation d'une pile de fenêtres.

    Entrées         : ST : La pile à initialiser.

    Préconditions   : ST n'existe pas.

    Sorties         : ST : La pile.

    Postconditions : ST est initialiser.
)
egin
  st := nil;
nd;


rocedure invw(hg,bd : point);
*   OBJECTIF        : Inverse une fenêtre définie par hg et bd.

    Entrées         : HG, BD : Description de la position de la fenêtre.

    Postconditions  :  A  l'écran la  fenêtre définie  est  mise  en  inversé :
aractère Noir sur fond Blanc.
)
ar l,c : integer;
egin
    for l := hg.l to bd.l do
        for c:= hg.c to bd.c do
           mem[adec : 1+2*(c-1+l*80-80)] := inv ;
nd;

rocedure truew(hg,bd : point);
*   OBJECTIF        : Met en normal un fenêtre définie par hg et bd.

    Entrées         : HG, BD : Description de la position de la fenêtre.

    Postconditions :  La fenêtre est mise en mode normal, caractères clairs sur
ond noir.
)
ar l,c : integer;
egin
    for l := hg.l to bd.l do
        for c:= hg.c to bd.c do
           mem[adec : 1+2*(c-1+l*80-80)] := vrai;
nd;


ROCEDURE INVVIDEO;
*   OBJECTIF        : Mise du vidéo en mode inverse.
)
```

```
EGIN
   TEXTCOLOR(0);TEXTBACKGROUND(15);
ND;

ROCEDURE TRVIDEO;
*   OBJECTIF        : Mise du vidéo en mode normal.
)
EGIN
   TEXTCOLOR(15);TEXTBACKGROUND(0);
ND;

rocedure bordsimple(hg,bd : point);
*   OBJECTIF        : Mettre un bord sur la fenêtre décrite par hg et bd.

    Entrées         : HG, BD : Description de la position de la fenêtre.

    Postconditions : La fenêtre est affichée avec un bord.
)
ar l,c,ofse : integer;
egin
 mem[adec : 2 * (hg.c + hg.l * 80 - 81)] := 218;
 mem[adec : 2 * (hg.c + bd.l * 80 - 81)] := 192;
 mem[adec : 2 * (bd.c + hg.l * 80 - 81)] := 191;
 mem[adec : 2 * (bd.c + bd.l * 80 - 81)] := 217;
 for c := hg.c + 1 to bd.c - 1 do
 begin
   mem[adec : 2 * (c + hg.l * 80 - 81)] := 196;
   mem[adec : 2 * (c + bd.l * 80 - 81)] := 196;
 end;
 for l := hg.l+1 to bd.l - 1 do
 begin
   ofse := (L - 1) * 160;
   mem[adec : ofse ] := 179;
   mem[adec : ofse + 158] := 179;
 end;
nd;
```

ontenu

Ce niveau contient les procédures :
- d'affichage de messages d'erreurs.
- de gestion de fichier texte.
- d'affichage d'écrans d'aide.

REMARQUES :
Postconditions sur les procédures fournissant er : En cas d'erreur, er est <> 0 et contient un code décrit dans le manuel du TURBO.

rogramme et spécifications.

```
*------------------------------------------------------------------------*)
*                                                                        *)
*          ***********      NIVEAU   50              ***********        *)
*                                                                        *)
*------------------------------------------------------------------------*)
*  >>>>>>>>>>>>>>>>>>>>>>>>     ERREUR      <<<<<<<<<<<<<<<<<<<<<<<<<<<<  *)
*  ┌──────────────────────────────────────────────────────────────────┐  *)
*  I NIVEAU 56                                                        I  *)
*  I                          ERREUR                                  I  *)
*  I                                                                  I  *)
*  └──────────────────────────────────────────────────────────────────┘  *)
```

```
rocedure erreur(er : integer);
*   OBJECTIF          :  Affichage dans le bas de l'écran  d'un message
'erreur (ligne 24).

   Entrées           : ER : Le code du message d'erreur.

   Préconditions  :   Si   er  n'est  pas  repertorié,   un  message
articulier apparaît.

   Postconditions  :  Le message disparaît dés  que  l'utilisateur a
apé sur une touche.

ar l :ligne ;
  hg, bd : point;

egin
 hg.l := 24 ; hg.c := 1;
 bd.l := 25 ; bd.c := 80;
 pushw(HG,BD,st);
 clsw(hg,bd);
 case er of
   1 : l := '                Erreur lors de l''accés aux fichiers ';
   2 : l := '                   Votre Date n''est pas valide ';
  10 : l := '                   Il n''existe aucun patient ';
  11 : l := '           Erreur Au Niveau de la Gestion du Fichier Patient ';
  else l := '                   Erreur non encore répertoriée ';
 end;
 gotoxy(1,24);
 writeln(l);
 Beep;
 gotoxy(50,25);
 write('TAPEZ UNE TOUCHE');
 bd.l := 24;
 invw(hg,bd);
```

```
   wait;
   popw(st);
end;

(*  >>>>>>>>>>>>>>>>>>>>>>>>  FICHIER TEXTE   <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(* ┌──────────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 54                                                               I *)
(* I                         FICHIER TEXTE                                   I *)
(* I                                                                         I *)
(* └──────────────────────────────────────────────────────────────────────────┘ *)
procedure debut_fit(fichier : nomfichier; var er :integer);
(*   OBJECTIF         : Ouverture d'un fichier.

     Entrées          : FICHIER : Le nom du fichier à ouvrir.

     Sorties          : F : variable globale représentant le fichier.
                        ER : Le code d'erreur décrivant l'accès.

     Postconditions :  Si er  est <>  0,  un message affiche le nom du
fichier présentant des difficultés.
*)
begin
    if vers = floppy then fichier := 'b:' + fichier;
    assign(f,fichier);
    {$I-} reset(f); {$I+}
    er := ioresult;
    if er <> 0 then
    begin
      gotoxy(1,10);
      write('erreur sur le fichier -- : ',fichier);
      erreur(1);
    end;
end;


procedure lect_fit(var l : ligne ; var er :integer);
(*   OBJECTIF         : Lecture d'une ligne du fichier texte F.

     Entrées          : F : le fichier que l'on doit lire.

     Préconditions    : Le fichier F doit avoir été ouvert.

     Sorties          : L : la ligne qui vient d'être lue.
                        ER : Le code d'erreur décrivant l'accès.
*)
begin
    {$I-} readln(f,l); {$I+}
    er := ioresult;
    if er <> 0 then erreur(1);
end;


procedure fin_fit(var er : integer);
(*   OBJECTIF         : Fermeture du fichier F.

     Entrées          : F : le fichier à fermer.

     Préconditions    : Le fichier doit avoir été ouvert.
```

```
     Sorties          :  ER :  un code d'erreur décrivant comment s'est
passé la fermeture du fichier.

     Postconditions : F est fermé.
*)
begin
    {$I-} close(f); {$I+}
    er := ioresult;
    if er <> 0 then erreur(1);
end;


(*  >>>>>>>>>>>>>>>>>>>>>>>>>>       AIDE          <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(*  ┌---------------------------------------------------------------------------┐ *)
(*  I NIVEAU 52                                                               I *)
(*  I                              AIDE                                       I *)
(*  I                                                                         I *)
(*  └---------------------------------------------------------------------------┘ *)
procedure me_aide(numaide : refaide);
(*   OBJECTIF           :  Affiche  à l'écran le  fichier correspondant à     .
numaide.

     Entrées           : numaide : Numéro de l'écran d'aide à afficher.

     Préconditions   : Numaide doit correspondre à un fichier existant.

     Postconditions : Le fichier est affiché à l'écran.
*)
var fichier : nomfichier ;
    l : ligne ;
    poshelp,hg,bd : point;
    er : integer;
begin
  fichier := 'help'+numaide+'.hlp';
  debut_fit(fichier,er);
  poshelp.l := 3 ; poshelp.c := 3;
  hg.l := 1; hg.c := 1;
  bd.l := 25 ; bd.c := 80;
  pushw(hg,bd,st);
  clrscr;
  bd.l := 24; (* à cause de cachecurs *)
  bordsimple(hg,bd);
  lect_fit(l,er);(* ligne de commentaire *)
  lect_fit(l,er);
  while l[me_prem] <> '.' do
  begin
     gotoxy(poshelp.c,poshelp.l);
     write(l);
     lect_fit(l,er);
     poshelp.l := poshelp.l + 1;
  end;
  fin_fit(er);
  wait;
  popw(st);
end;
```

Contenu

Ce niveau contient les procédures de lecture au clavier de données typées:

- Lecture d'une chaîne de caractères.
- Lecture d'une date.
- Lecture d'une heure.
- lecture d'une entier.
- Lecture d'un réel.
- Lecture de oui ou non.

REMARQUES : Une date peut se présenter sous deux formats:
- Normal : JJMMAAAA
- Extérieur : AAAAMMJJ.

Procédures et spécifications

```
(*-------------------------------------------------------------------------*)
(*                                                                         *)
(*          ***********      NIVEAU   40              ***********         *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)
(*  >>>>>>>>>>>>>>>>>>>>>>  OUTILS LECTURE  <<<<<<<<<<<<<<<<<<<<<<<<<<<<  *)
(*  ┌-----------------------------------------------------------------┐    *)
(*  I NIVEAU 48                                                       I    *)
(*  I                        OUTILS LECTURE                           I    *)
(*  I                                                                 I    *)
(*  └-----------------------------------------------------------------┘    *)
procedure lc_chaine(ou : point ; long : integer ;
                    sc : sofc  ; sctrl : sofctrl;
                    var c : char ; var tc : touchectrl; var result : chainec;
                    naide : refaide);

(*   OBJECTIF        :  Lecture d'une chaîne de caractères.  Les
touches de 'contrôle' utilisables sont : DELETE, DEL, GAUCHE, DROITE +
celles définies dans SCTRL.

     Entrées         : OU : L'endroit où doit s'effectuer la lecture.
                       SC   :  L'ensemble  des  caractères  normaux
acceptables.
                       SCTRL :  L'ensemble des  caractères  de contrôle
acceptables.
                       NAIDE  :  Le numéro  de l'écran  d'aide qui doit
apparaître si l'utilisateur tape sur F1.

     Sorties         : C : Le dernier caractère normal tapé.
                       TC : Le dernier caractère de contrôle tapé.
                       RESULT : La chaîne de caractères qui a été lue.

     Postconditions : Si le dernier caractère frappé par l'utilisateur
est un normal TC contient 'autre'.
*)
var posi : integer;
    fin  : boolean;
    i    : integer;
begin
   posi := length(result) + 1;
   for i := posi to long do result := result + ' ';
```

```
   fin := false;
   posi := ou.c;
   gotoxy(ou.c,ou.l);
   write(result);
   gotoxy(ou.c,ou.l);
   repeat
     case tc of
          DELETE      : If (posi > ou.c) then
                        begin
                             posi := posi - 1;
                             result[posi - ou.c + 1] := ' ';
                             gotoxy(posi,ou.l);
                             write(' ');
                             gotoxy(posi,ou.l);
                        end;
          DEL         : begin
                             result[posi - ou.c + 1] := ' ';
                             gotoxy(posi,ou.l);
                             write(' ');
                             gotoxy(posi,ou.l);
                        end;
          GAUCHE      : if posi > ou.c then
                        begin
                           posi := posi - 1 ;
                           gotoxy(posi,ou.l);
                        end;
          DROITE      : if posi <= ou.c + long - 1 then
                        begin
                           posi := posi + 1 ;
                           gotoxy(posi,ou.l);
                        end
                        else if posi = ou.c + long - 1
                             then gotoxy(posi,ou.l);
          F1          : me_aide(naide);
          AUTRE       : if (posi <= ou.c + long - 1) and ( c in sc )then
                        begin
                          result[posi - ou.c + 1] := c;
                          posi := posi + 1 ;
                          write(c);
                        end
     end;
     if tc in sctrl then fin := true
                    else lect(sc,sctrl+[F1,DEL,DELETE,GAUCHE,DROITE],c,tc);
  until fin;
end;
(*   >>>>>>>>>>>>>>>>>>>>>>>>   OUTILS DATE    <<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(*  ┌-----------------------------------------------------------------------┐ *)
(*  I NIVEAU 44                  OUTILS DATE                              I *)
(*  └-----------------------------------------------------------------------┘ *)
procedure date_ext(var d : date);
(*   OBJECTIF         : Convertit le format AAAAMMJJ en JJMMAAAA.

     Entrées          : d : la date dans le format AAAAMMJJ.

     Sorties          : d : la date sous le nouveau format.

     Postconditions : Aucune vérification de validité n'est faite.
*)
var d1,d2,d3,d4 : char;
begin
```

```
   d1   := d[1];   d2 := d[2] ; d3 := d[3] ; d4 := d[4];
   d[1] := d[7]; d[2] := d[8] ; (* jour *)
   d[3] := d[5]; d[4] := d[6] ; (* mois *)
   d[5] := d1  ; d[6] := d2   ; d[7] := d3 ; d[8] := d4; (* année *)
end;

procedure ext_date(var d : date);
(*   OBJECTIF        : Convertit le format JJMMAAAA en AAAAMMJJ.

     Entrées         : d : la date dans le format JJMMAAAA.

     Sorties         : d : la date sous le nouveau format.

     Postconditions : Aucune vérification de validité n'est faite.
*)
var d1,d2,d3,d4 : char;
begin
   d1   := d[1] ;   d2 := d[2] ; d3 := d[3] ; d4 := d[4];
   d[1] := d[5] ; d[2] := d[6]  ; d[3] := d[7] ; d[4] := d[8];(* année *)
   d[5] := d3   ; d[6] := d4 ; (* mois *)
   d[7] := d1   ; d[8] := d2 ; (* jour *)
end;

procedure lec_datj(var d : date);
(*   OBJECTIF        :  Lecture de la date du jour sur le calendrier de
l'ordinateur.

     Sorties         : d : La date du jour.

     Postconditions : Le format de la date : AAAAMMJJ.
*)

type
      regpack = record
                ax,bx,cx,dx,bp,di,si,ds,es,flags: integer;
                end;
      genre = (chif,cara);
      truc = record case genre of
                    chif : ( val   :integer);
                    cara : ( c1,c2 : char);
             end;

var recpack : regpack;
    mois,jour : string[2];
    annee     : string[4];
    truk      : truc;

begin
    d := 'aaaammjj';
    with recpack do
    begin
        ax := $2a shl 8;
    end;
    msdos(recpack);
    with recpack do
    begin
        str(cx,annee);
        str(dx mod 256, jour);
        str(dx shr 8,mois);
    end;
```

```
    if length(jour) = 1 then jour := '0' + jour;
    if length(mois) = 1 then mois := '0' + mois;
    d[1] := annee[1];
    d[2] := annee[2];
    d[3] := annee[3];
    d[4] := annee[4];
    d[5] := mois[1];
    d[6] := mois[2];
    d[7] := jour[1];
    d[8] := jour[2];
end;

function vali_dat(d : date): boolean;
(*   OBJECTIF        : Test de la validité de la date format extérieur !!!.

     Entrées         : D : La date à vérifier.

     Préconditions   : D est sous le format extérieur.

     Sorties         : Valid_dat : True : La date est valide.
                                   False : la date n'est pas valide.
*)

var code, j, m , a : integer;
    bon : boolean;
    R100,R4 : real;

begin
    bon := true;
    (* année de preman à dernan *)
    val(copy(d,5,4),a,code);
    if (a < preman) or (a > dernan)
       then bon := false;
    if bon then (* test du mois *)
    begin
        val(copy(d,3,2),m,code);
        if (m < 1) or (m > 12) then bon := false;
    end;
    if bon then (* test du jour *)
    begin
        val(copy(d,1,2),j,code);
        case m of
             4,6,9,11 : if (j < 1) or (j > 30 ) then bon := false;
             2        : begin
                          if (j < 1 ) or (j > 28) then
                          begin
                              if j = 29 then (* année bisextile *)
                              begin
                                R100 := FRAC(a / 100);
                                R4   := FRAC(a / 4);
                                if ((R4 <> 0) or (R100 = 0)) and (R4*R100 <> 0)
                                  then bon := false;
                              end
                              else bon := false
                          end
                        end;
                  else if(j<1) or (j > 31) then bon := false;
        end;
     end;
     if not bon then erreur(2);
```

```
   vali_dat := bon;
end;

procedure avanc_dat(ou : point; d :date;var p : integer ; var pd : integer);
(*   OBJECTIF        : Avance d'un caractère dans la date.

     Entrées         : OU : L'endroit où doit se lire la date.
                       D  : La date par défaut.
                       P  : Pointeur à l'écran.
                       PD : Pointeur dans le tableau date.

     Préconditions   : PD doit être compris entre 1 et 6.

     Sorties         : P et PD sont mis à jour.
*)

begin
  pd := pd + 1;
  case p of
       1,4,7,8,9    : p := p + 1;
       2,5          : p := p + 2;
       10           : begin
                         p := 1;
                         pd := 1;
                      end;
  end;
  gotoxy(ou.c+p-1,ou.l);
  write(d[pd]);
  gotoxy(ou.c+p-1,ou.l);
end;

procedure recul_dat(ou : point; d : date ;var p : integer ; var pd : integer);
(*   OBJECTIF        : Recule d'un caractère dans la date affichée.

     Entrées         : OU : L'endroit où doit se lire la date.
                       D  : La date par défaut.
                       P  : Pointeur à l'écran.
                       PD : Pointeur dans le tableau date.

     Préconditions   : PD doit être compris entre 1 et 6.

     Sorties         : P et PD sont mis à jour.

begin
  pd := pd -1;
  case p of
       1            : begin p := 1; pd := 1 end;
       2,5,8,9,10   : p := p - 1;
       4,7          : p := p - 2;
  end;
  gotoxy(ou.c+p-1,ou.l);
  write(d[pd]);
  gotoxy(ou.c+p-1,ou.l);
end;
(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 42                  OUTILS DATE                               │ *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure lec_dat(ou : point ; sctrl : sofctrl
              ;var d : date ;var c : char;var tc : touchectrl;
              naide : refaide);
```

```
(*   OBJECTIF          : Lecture d'une date à l'écran.

     Entrées           : OU : L'endroit où doit s'effectuer la lecture.
                         SCTRL :  L'ensemble des  caractères  de contrôle
admissible en plus des habituels (ESC, DEL,GAUCHE...)
                         NAIDE :  Le numéro de l'écran d'aide  à afficher
si l'on utilise la touche F1.

     Sorties           : DATE : la date lue.
                         C  : Le dernier caractère normal lu.
                         TC : Le dernier cacaractère de contrôle lu.
*)

var p,pd : integer; (* position *)
    fin : boolean;

begin
    date_ext(d);
    tc := F10;
    c := '#';
    p := 1; pd := 1;
    fin := false;
    gotoxy(ou.c,ou.l);
    while not fin do
    begin
    case tc of
              gauche : if p = 1 then fin := true
                                else recul_dat(ou,d,p,pd);
              droite : if p = 10 then fin := true
                                else avanc_dat(ou,d,p,pd);
              F1     : me_aide(naide);
              autre  : begin
                             d[pd] := c;
                             write(d[pd]);
                             if p = 10 then fin := vali_dat(d);
                             avanc_dat(ou,d,p,pd);
                       end;
             else if tc in sctrl then fin := true;
    end; (* case *)
    if fin
    then begin fin := vali_dat(d);
               p := 10;
               avanc_dat(ou,d,p,pd);
         end;
    if not fin
       then lect(['0'..'9'],sctrl+[F1,gauche,droite],c,tc);
 end;
 ext_date(d);
end;

function accept_date(inf,sup,d : date): boolean;
(*   OBJECTIF          : Test si la date est comprise entre INF et SUP.
*)
begin
 accept_date := (inf <= d) and (d <= sup);
end;
(*  ┌──────────────────────────────────────────────────────────────────┐ *)
(*  I NIVEAU 40                                                        I *)
(*  I                          OUTILS DATE                             I *)
```

```
(* I                                                                    I *)
(* L--------------------------------------------------------------------J *)
procedure lc_date(ou : point ; inf,sup : date;sctrl : sofctrl ;
                  var c : char  ; var tc : touchectrl ; var d : date;
                  naide : refaide);
(*   OBJECTIF        : Lecture d'une date au clavier avec affichage de
celle-ci  à la  position  OU,  plus vérification de la  validité de la
date.

     Entrées         : OU : L'endroit où doit s'afficher la date.
                       INF :  La date  lue ne  peut  être  antérieure à
cette date.
                       SUP :  La date  lue ne  peut  être postérieure à
cette date.
                       SCTRL  :  Ensemble  des  touches  de  'contrôle'
acceptable.
                       NAIDE :  Numéro de l'écran d'aide  à afficher si
l'utilisateur tape sur F1.

     Préconditions   : OU doit être une valeur comprise dans l'écran.
INF ≤ SUP et doivent être des dates respectant le format extérieur.

     Sorties         : D : La date lue au clavier.
                       C : Le dernier caractère normal lu.
                       TC : Le dernier caractère de contrôle lu.

     Postconditions :  Si le dernier caractère lu est  normal alors TC
contient la valeur 'autre'.
*)

var fin : boolean;
begin
  fin := false;
  repeat
    lec_dat(ou,sctrl,d,c,tc,naide);
    if accept_date(inf,sup,d) then fin := true
                              else erreur(100);
  until fin;
end;

procedure aff_date(ou : point ; d : date);
(*   OBJECTIF        : Affichage d'une date à la position OU.

     Entrées         : OU : La position où la date doit être affichée.
                       D : la date à afficher.

     Préconditions   : D doit être écrit dans le format normal.

     Postconditions : La date D est affichée à la position OU.
*)

begin
    gotoxy(ou.c,ou.l);
    write(d[7],d[8],'/',d[5],d[6],'/',d[1],d[2],d[3],d[4]);
    cachecurs;
end;

procedure initdate;
(*   OBJECTIF        :  Initialise les variables  globales PREMDATE  et
INITDATE à la date du jour.
```

```
     Entrées         :  PREMDATE, DERNDATE : Variables globales de type
date.

     Sorties         :  PREMDATE  et  DERNDATE  contiennent la  date du
jour.

     Postconditions : Les variables dates sont au format normal.
*)

begin
   lec_datj(datejour);
   premdate := datejour;
   derndate := datejour;
end;

procedure datemenu1;
(*   OBJECTIF        : Insèrer la date du jour dans le menu numéro 1.

     Postcondition :  Ceci s'effectue au niveau du  menu enregistré en
mémoire.
*)

var
    adme1      : tampon;
begin
(* mise dans le menu *)
    adme1 := tp[me_prem].pme;
    mem[seg(adme1^):ofs(adme1^)+70] := ord(datejour[7]);
    mem[seg(adme1^):ofs(adme1^)+71] := ord(datejour[8]);
    mem[seg(adme1^):ofs(adme1^)+73] := ord(datejour[5]);
    mem[seg(adme1^):ofs(adme1^)+74] := ord(datejour[6]);
    mem[seg(adme1^):ofs(adme1^)+76] := ord(datejour[1]);
    mem[seg(adme1^):ofs(adme1^)+77] := ord(datejour[2]);
    mem[seg(adme1^):ofs(adme1^)+78] := ord(datejour[3]);
    mem[seg(adme1^):ofs(adme1^)+79] := ord(datejour[4]);
end;
(*  >>>>>>>>>>>>>>>>>>>>>>>>>>>   OUTILS HEURE   <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(* ┌---------------------------------------------------------------------------┐ *)
(* │ NIVEAU 46                      OUTILS HEURE                               │ *)
(* └---------------------------------------------------------------------------┘ *)

function accept_heure(inf,sup,h : heure) : boolean;
(*   OBJECTIF        :  Teste si une  date est valide et  comprise entre
les bornes INF et SUP.

     Entrées        : INF : La borne inférieure.
                      SUP : La borne supérieure.
                      H   : La date à tester la validité.

     Précondition : INF et SUP sont des heures valides.

     Sorties       : True : H est valide et compris entre INF et SUP.
                     False sinon.
*)

begin
    accept_heure := (inf <= h) and (h <= sup);
end;
(* ┌---------------------------------------------------------------------------┐ *)
```

```
(* I NIVEAU 44                    OUTILS HEURE                                I *)
(* L_____________________________________________________________________J *)
procedure time(var h : heure);
(*   OBJECTIF        : Lecture de l'horloge de l'ordinateur.

     Sorties         : H : l'heure qui vient d'être lue.
*)

type timestring = string[11];
     regpack = record
               ax,bx,cx,dx,bp,di,si,ds,es,flags: integer;
               end;

var recpack : regpack;
    ah,al,ch,cl,dh : byte;
    hour,min,sec,cent : string[2];

begin
    ah := $2c;
    with recpack do
    begin
        ax := ah shl 8 + al;
    end;
    intr($21, recpack);
    with recpack do
    begin
        str(cx shr 8, hour);
        str(cx mod 256, min);
        str(dx shr 8,sec);
        str(dx mod 256, cent);
    end;
    h :=hour+min+sec;
end;

function vali_heure(h : heure): boolean;
(*   OBJECTIF        : Test de la validité de l'heure format extérieur !!!

     Entrées         : H : l'heure à tester.

     Sorties         : True : l'heure est valide.
                       False : sinon.
*)

var he,mi,sec : string[6];
   ok : boolean;
begin
   he := h ; mi := h ; sec := h;
   delete(he,3,4);
   delete(mi,1,2);
   delete(mi,3,2);
   delete(sec,1,4);
   ok := ('00' <= he) and (he <= '24') and
         ('00' <= mi) and (mi <= '59') and
         ('00' <= sec) and (sec <= '59');
   if not ok then erreur(100);
   vali_heure := ok;
end;


procedure avanc_heure(ou : point; h : heure;var p : integer ; var pd : integer);
```

```
*   OBJECTIF        : Avance d'un caractère dans l'heure affichée.

    Entrées         :  OU  :  L'endroit  du  premier caractère  où est
affiché l'heure.
                       D  : La date par défaut.
                       P  : Pointeur à l'écran.
                       PD : Pointeur dans le tableau date.

    Préconditions   : PD doit être compris entre 1 et 6.

    Sorties         : P et PD sont mis à jour.
)

begin
 pd := pd + 1;
 case p of
      1,4      : p := p + 1;
      2,5      : p := p + 2;
      7        : begin
                     p := 1;
                     pd := 1;
                 end;
 end;
 gotoxy(ou.c+p-1,ou.l);
end;

procedure recul_heure(ou : point; h:heure ;var p : integer ; var pd : integer);
*   OBJECTIF        : Recule d'un caractère dans l'heure affichée.

    Entrées         : OU  :  L'endroit  du  premier  caractère  où est
affiché l'heure.
                       D  : La date par défaut.
                       P  : Pointeur à l'écran.
                       PD : Pointeur dans le tableau date.

    Préconditions   : PD doit être compris entre 1 et 6.

    Sorties         : P et PD sont mis à jour.
)

egin
 pd := pd - 1;
 case p of
      1        : begin p := 1; pd := 1 end;
      2,5      : p := p - 1;
      4,7      : p := p - 2;
 end;
 gotoxy(ou.c+p-1,ou.l);
nd;
*  ┌-------------------------------------------------------------------------┐ *)
*  I NIVEAU 42                  OUTILS HEURE                                I *)
*  └-------------------------------------------------------------------------┘ *)
rocedure lec_heure(ou : point ; sctrl : sofctrl
             ;var h : heure ;var c : char;var tc : touchectrl ;
             naide : refaide);
    OBJECTIF        : Lecture d'une heure au clavier.

    Entrées         : OU : L'endroit où l'heure doit être lue.
                       SCTRL  :  L'ensemble  des  touches  de contrôles
ceptable.
```

```
                    NAIDE :  Le numéro de l'écran d'aide  à afficher
si l'utilisateur tape sur F1.

    Sorties         : C: La dernière touche 'normale' lue.
                      TC : La dernière touche de 'contrôle' lue.
                      H  : L'heure qui vient d'être lue.

    Postconditions  :  TC  contient  'autre'  si  la  dernière touche
frappée est un caractère normal.
*)

var p,pd : integer; (* position *)
    fin : boolean;

begin
    tc := F10;
    c := '#';
    p := 1; pd := 1;
    fin := false;
    gotoxy(ou.c,ou.l);
    while not fin do
    begin
    case tc of
              gauche : if p = 1 then fin := true
                                else recul_heure(ou,h,p,pd);
              droite : if p = 7 then fin := true
                                 else avanc_heure(ou,h,p,pd);
              F1     : me_aide(naide);
              autre  : begin
                             h[pd] := c;
                             write(c);
                             if p = 7 then fin := vali_heure(h);
                             avanc_heure(ou,h,p,pd);
                       end;
             else if tc in sctrl then fin := true;
    end; (* case *)
    if fin
    then begin fin := vali_heure(h);
               p := 7;
               avanc_heure(ou,h,p,pd);
         end;
    if not fin
       then lect(['0'..'9'],sctrl+[F1,gauche,droite],c,tc);
 end;
end;

* ┌──────────────────────────────────────────────────────────────────────┐ *)
* I NIVEAU 40                                                            I *)
* I                          OUTILS HEURE                                I *)
* I                                                                      I *)
* └──────────────────────────────────────────────────────────────────────┘ *)
rocedure lc_heure(ou : point ; inf,sup : heure ;sctrl : sofctrl ;
             var c : char  ; var tc : touchectrl ; var h : heure ;
             naide : refaide);
*   OBJECTIF        :  Lecture de l'heure au clavier avec vérification
e validité.

    Entrées         : OU : l'endroit où doit être lue la date.
                      INF,  SUP : Borne inférieure et supérieure entre
esquelles l'heure doit être comprise pour être valide.
```

```
                      NAIDE :  Le numéro de l'écran  d'aide à afficher
si l'utilisateur tape sur F1.

     Préconditions   : INF doit être ≤ à SUP.

     Sorties         : C: La dernière touche 'normale' lue.
                       TC : La dernière touche de 'contrôle' lue.
                       H  : L'heure qui vient d'être lue.

     Postconditions  :  TC  contient  'autre'  si  la  dernière touche
frappée est un caractère normal.
*)

var fin : boolean;
begin
  fin := false;
  repeat
    lec_heure(ou,sctrl,h,c,tc,naide);
    if accept_heure(inf,sup,h) then fin := true
                               else erreur(100);
  until fin;
end;

procedure aff_heure(ou : point ; h : heure);
(*   OBJECTIF        : Affiche une heure à l'endroit désigné par OU.

     Entrées         : OU : L'endroit où l'on doit afficher l'heure.
                       H : L'heure à afficher.
     Postcondition : Une fois la date affichée le curseur est caché.
*)

begin
    gotoxy(ou.c,ou.l);
    write(h[1],h[2],':',h[3],h[4],':',h[5],h[6]);
    cachecurs;
end;

(*  »»»»»»»»»»»»»»»»»»»»»»» LECTURE ENTIER   «««««««««««««««««««««««««««««« *)
(* ┌────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 42                 LECTURE ENTIER                           │ *)
(* └────────────────────────────────────────────────────────────────────┘ *)
procedure tue_blancs(var cdc : chainec);
(*   OBJECTIF        :  Retire les caractères  blancs  de  cdc, ce qui
revient à aligner la chaîne de caractères à gauche.

     Entrées         : CDC : la chaîne à modifier.

     Sorties         : CDC : la chaîne alignée à gauche.
*)

var i : integer;
    l : integer;
    fin : boolean;
begin

   l := length(cdc);
   (* blancs avants *)

   i := 1;  fin := false;
   while (i <= l) and not fin do
```

```
        if cdc[i] = ' ' then i := i+1
                        else fin := true;
   i := i - 1;
   delete(cdc,1,i);
   (* caractère aprés un blancs jusqu'a la fin *)
   l := length(cdc);
   i := 1; fin := false;
   while (i <= l) and not fin do
        if cdc[i] <> ' ' then i := i + 1
                        else fin := true;
   delete(cdc,i,l-i+1);
end;

function valid_ent(inf,sup,res,code : integer) : boolean;
(*   OBJECTIF        :  Test si  un  entier  est valide,  il  doit être
compris entre INF et SUP et CODE doit être = 0.

     Entrées         : INF  et  SUP :  borne  inférieure  et supérieure
entre lesquelles l'entier doit être.
                       RES : L'entier à tester.
                       CODE : Code d'erreur.

     Sorties         : True  : Res est valide.
                       False sinon.
*)
begin
   valid_ent := (code = 0 ) and (inf <= res ) and ( res <= sup);
end;
(* ┌------------------------------------------------------------------------┐ *)
(* I NIVEAU 40                                                              I *)
(* I                          LECTURE ENTIER                                I *)
(* I                                                                        I *)
(* └------------------------------------------------------------------------┘ *)
procedure lc_entier(ou : point; inf,sup : integer ; sctrl : sofctrl;
          var c : char ; var tc : touchectrl ; var result : integer;
          naide : refaide);
(*   OBJECTIF        :  Lecture d'un entier au clavier et  affichage de
ce dernier au point OU.   De plus l'entier subit un  test de validité,
i.e. on vérifie s'il est compris entre les borne INF et SUP.

     Entrées         : OU : L'endroit où l'entier doit être lu.
                       INF,  SUP :  les bornes inférieure et supérieure
entre lesquelles l'entier doit être compris.
                       NAIDE :  Le numéro de l'écran  d'aide à afficher
si l'utilisateur tape sur F1.

     Préconditions   : INF doit être ≤ à SUP.

     Sorties         : C: La dernière touche 'normale' lue.
                       TC : La dernière touche de 'contrôle' lue.
                       RESULT : L'entier qui vient d'être lu.

     Postconditions  :  TC  contient  'autre'  si  la  dernière touche
frappée est un caractère normal.
*)

const long = 5 ; (* nbre max de caractères composant un entier *)
var cdc : chainec;
    fin : boolean;
    code : integer;
```

```
    i     : integer;
begin
   fin := false;
   repeat
      str(result,cdc);
      tc := F10;
      lc_chaine(ou,long,[' ','0'..'9'],sctrl,c,tc,cdc,naide);
      tue_blancs(cdc);
      val(cdc,result,code);
      if valid_ent(inf,sup,result,code)
         then fin := true
         else erreur(100);
  until fin;
   str(result,cdc);
   tue_blancs(cdc);
   code := length(cdc);
   gotoxy(ou.c,ou.l);
   write(cdc);
   for i := 1 to 79 - ou.c - code do write(' ');
end;

(*   ››››››››››››››››››››››››› LECTURE REEL      ‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹‹ *)
(* ┌──────────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 42                     LECTURE REEL                              I *)
(* └──────────────────────────────────────────────────────────────────────────┘ *)
function valid_reel(inf,sup,res: real ; code : integer) : boolean;
(*   OBJECTIF          : Test si un réel est valide, il doit être compris
entre INF et SUP et CODE doit être = 0.

     Entrées           : INF  et  SUP :  borne  inférieure  et supérieure
entre les quelles le réel doit être.
                         RES : L'entier à tester.
                         CODE : Code d'erreur.

     Sorties           : True   : Res est valide.
                         False sinon.
*)
begin
   valid_reel := (code = 0 ) and (inf <= res ) and ( res <= sup);
end;

procedure ec_reel(ou: point ; r :real);
(*   OBJECTIF          : Affichage d'une réel à la position OU.

     Entrées           : OU : La position où le réel doit être affiché.
                         R : Le réel à afficher.

     Postconditions : Le réel R est affiché à la position OU.
*)
var cdc:chainec;
    i,l : integer;
begin
   str(r:37:4,cdc);
   tue_blancs(cdc);
   l := length(cdc);
   gotoxy(ou.c,ou.l);
   write(cdc);
   for i := 1 to 79 - ou.c - l do write(' ');
end;
(* ┌──────────────────────────────────────────────────────────────────────────┐ *)
```

```
(* I NIVEAU 40                                                          I *)
(* I                          LECTURE REEL                              I *)
(* I                                                                    I *)
(* L--------------------------------------------------------------------J *)
procedure lc_reel(ou : point; inf,sup : real ; sctrl : sofctrl;
          var c : char ; var tc : touchectrl ; var result : real;
          naide : refaide);
(*   OBJECTIF        :  Lecture d'un réel au clavier et affichage de ce
dernier au point OU.  De plus le réel  subit un test de validité, i.e.
on vérifie s'il est compris entre les borne INF et SUP.

     Entrées         : OU : L'endroit où le réel doit être lu.
                       INF,  SUP :  les bornes inférieure et supérieure
entre lesquelles l'entier doit être compris.
                       NAIDE :  Le numéro de l'écran  d'aide à afficher
si l'utilisateur tape sur F1.

     Préconditions   : INF doit être ≤ à SUP.

     Sorties         : C: La dernière touche 'normale' lue.
                       TC : La dernière touche de 'contrôle' lue.
                       RESULT : Le réel qui vient d'être lu.

     Postconditions  :  TC  contient  'autre'  si  la  dernière touche
frappée est un caractère normal.
*)


const long = 42 ; (* nbre max de caractères composant un réel *)
var cdc : chainec;
    fin : boolean;
    code : integer;

begin
   fin := false;
   repeat
      str(result:37:4,cdc);
      tc := F10;
      tue_blancs(cdc);
      lc_chaine(ou,long,[' ','.','e','E','0'..'9'],sctrl,c,tc,cdc,naide);
      tue_blancs(cdc);
      val(cdc,result,code);
      if valid_reel(inf,sup,result,code)
         then fin := true
         else erreur(100);
  until fin;
  ec_reel(ou,result);
end;

(* ,--------------------------------------------------------------------, *)
(* I NIVEAU 40                                                          I *)
(* I                        LECTURE OUI NON                             I *)
(* I                                                                    I *)
(* L--------------------------------------------------------------------J *)
procedure lc_ouinon(ou : point ; sctrl :sofctrl ;
                    var res : boolean ; var tc : touchectrl ;
                    naide : refaide);
(*   OBJECTIF        : Lecture au clavier des touches 'Oo' ou 'Nn' pour
oui et pour non.
```

```
     Entrées          : OU : L'endroit où doit s'effectuer la lecture.
                        TC : La dernière touche de contrôle utilisée.
                        NAIDE :  Le numéro de l'écran  d'aide à afficher
en cas de demande.

     Sorties          :  RES :  Thrue :  Si l'utilisateur a tapé 'O' ou
'o'.
                                 False :  Si l'utilisateur a tapé 'N' ou
'n'.
*)

var hg,bd : point;
    c     : char;
begin
   hg := ou ; bd := ou;
   gotoxy(ou.c,ou.l);
   write('OUI NON');
   cachecurs;
   repeat
      truew(hg,bd);
      if res then begin hg.c := ou.c ; bd.c := ou.c + 2 end
             else begin hg.c := ou.c+4 ; bd.c := ou.c + 6 end;
      invw(hg,bd);
      lect(['O','o','N','n'],sctrl+[F1,GAUCHE,DROITE],c,tc);
      case tc of
           gauche,droite : res := not res;
           autre : case c of
                     'O','o' : res := true;
                     'N','n' : res := false;
                   end;
           F1      : me_aide(naide);

      end;
   until tc in sctrl;
   truew(hg,bd);
   gotoxy(ou.c,ou.l);
   if res then write('OUI     ')
          else write('NON     ');
   cachecurs;
end;
```

Contenu

Dans ce niveau on retrouve toutes les procédures permettant de:

- Lire les menus.
- Voyager dans un menu.
- D'afficher les écrans de démonstration.

Ce niveau dans la version définitive devait contenir toutes les procédures d'accès à la base de données.

NOTATION : On appelle
- 'Menu Vertical' un menu dans lequel on se déplace par un voyage vertical entre ses libellés.
- 'Menu horizontal' un menu dans lequel on se déplace par un voyage horizontal entre ses libellés.
- 'Menu Vertical & Horizontal' un menu dans lequel on se déplace par un voyage vertical et horizontal entre ses libellés.

REMARQUES :
- Toutes les procédures utilisant les menus présuposent que ces derniers existent.

- Toutes les procédures utilisant la pile de fenêtres présupposent son existence.

Procédures et spécifications

```
(*-------------------------------------------------------------------------*)
(*                                                                         *)
(*          ***********      NIVEAU   30              ***********         *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)
(*  >>>>>>>>>>>>>>>>>>>>>>>  LECT DES MENUS  <<<<<<<<<<<<<<<<<<<<<<<<<<<  *)
(*  ┌-----------------------------------------------------------------┐    *)
(*  ¦ NIVEAU 34                  LECT DES MENUS                       ¦    *)
(*  └-----------------------------------------------------------------┘    *)
procedure ligbuf(var b : tampon ; l : ligne);
(*   OBJECTIF          : Copie la ligne dans le buffer et remplace les
points par des blancs et si l'on trouve un # ce dernier est remplacé
par des blancs ainsi que les deux caractères qui le suivent *)

     Entrées           : L : la ligne à copier dans le buffer.

     Sorties           : B : Le buffer où l'on met la ligne.

     Postconditions : La ligne est ajoutée à la fin de B.
*)
var ic : integer ; (* indice de caractères *)
begin
   for ic := 1 to length(l) do
   begin
     if (l[ic] <> '.') and (l[ic] <> '@') and (l[ic] <> '#')
     then mem[seg(b^):ofs(b^)+ic-1] := ord(l[ic])
     else if l[ic] <> '#'
         then mem[seg(b^):ofs(b^)+ic-1] := 32
         else begin
```

```
                    mem[seg(b^):ofs(b^)+ic-1] := 32;
                    mem[seg(b^):ofs(b^)+ic]   := 32;
                    mem[seg(b^):ofs(b^)+ic+1] := 32;
                    ic := ic + 2;
                end;
    end;
    b := ptr(seg(b^),ofs(b^)+length(l));
end;

procedure bufmem(b : tampon ; combien : integer ; var ou : tampon);
(*   OBJECTIF        : Copie le buffer de 'combien' de bytes en mémoire
et fournit sa position en mémoire, i.e son adresse.

     Entrées         : B : Le buffer à mémoriser.
                       COMBIEN : La taille de ce buffer.

     Préconditions   : Aucune vérification n'est faite pour savoir s'il
y assez de place en mémoire.

     Sorties         : OU : L'endroit où se trouve le buffer.
*)

var i: integer;

begin
   getmem(ou,combien);
   for i := 0 to combien-1 do
      mem[seg(ou^):ofs(ou^) + i] := mem[seg(b^):ofs(b^) + i];
end;

procedure lirenuml( l : ligne ; var p : integer ; var result : integer);
(*   OBJECTIF        :  Lit  le  numéro  contenu dans la ligne  'l' et
commençant au caractère 'p'.
    Le numéro doit se  terminer par  un blanc,  par une virgule ou par
'/'.
   A la sortie p pointe sur le premier caractère après le nombre.

     Entrées         : L : La ligne où se trouve le numéro.
                       P : L'endroit où commence le nombre.

     Sorties         : P  :  Position du  premier  caractère  après le
nombre.
                       RES : le nombre.
*)

var i : integer;
    nbre : string[3]; (* max 999 *)
    ll   : integer;
begin
      ll := length(l);
      i := 1;  nbre := '  ';
      repeat
        nbre[i] := l[p];  i := i+1; p := p + 1;
      until (l[p] = ' ') or (l[p] =',') or (l[p] = '/') or (p > ll);
      val(nbre,result,i);
end;

procedure commentaire(var l :ligne ; var comm : tampon);
(*   OBJECTIF        : Lecture   d'une   ligne   de   commentaire,   et
mémorisation de celle-ci dans la mémoire.
```

```
     Entrées         : L : la ligne à mémoriser.

     Sorties         : COMM : l'endroit où la ligne est mémorisée.
*)

var er : integer;
begin
    getmem(comm,length(l)+1);
    move(l,comm^,length(l)+1);
    lect_fit(l,er);
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 32                  LECT DES MENUS                            │ *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure initme(var buffer : tampon);
(*   OBJECTIF        : Initialisation pour la  lecture des menus.
     - Ouverture du fichier contenant les écrans.
     - Création d'un tampon de la taille d'un écran.

     Sortie : BUFFER : Un espace de mémoire de la taille de l'écran.
*)

var  fichier : nomfichier;
     er      : integer;
begin
  fichier := 'ecran.'+ ecran_ext ;
  debut_fit(fichier,er);
  getmem(buffer,tecran);(* Créer un tampon de ‹la grandeur de l'écran *)
end;

procedure clotme(var buffer : tampon);
(*   OBJECTIF        : Clôture des actions de lecture des menus.

     Entrées         :  BUFFER :  Buffer ayant servi  à la mémorisation
des écrans.

     Préconditions   :  Le  fichier  des  écrans  est  considéré  comme
ouvert.

     Postconditions :  Le fichier des  écrans est fermé.  BUFFER a été
détruit, donc la place qu'on avait réservée est redevenue disponible.
*)

var er : integer;
begin
   fin_fit(er);(* fermeture du fichier *)
   freemem(buffer,tecran);(* détruire le tampon *)
   buffer := nil;
end;

procedure horiz(buffer : tampon ; var me : menu ; var l : ligne);
(*   OBJECTIF : Mémorisation d'un menu ME de type horizontal, i.e. que
l'on se déplace dedans par un voyage  à l'horizontal.

     Entrées  :  BUFFER :  L'espace mémoire où  l'on  doit  stocker le
menu.
                 L : La première ligne du menu lue sur le fichier.
```

```
     Sorties  : ME : le menu qui vient d'être lu et mémorisé.
                L : La ligne du menu suivant.
*)

var p : integer; (* position courante dans la ligne *)
    ll : integer ; (* longueur de la ligne courante *)
    ch   : pch ; (* pointeur vers le choix courant *)
    prec : pch ; (* pointeur vers le choix précedent *)
    debut : integer ; (* à partir d'où faut t'il inverser *)
    b     : tampon   ; (* fin du buffer, 1ere place libre *)
    num   : integer ; (* numéro du libellé courant *)
    er    : integer;
begin
    me.genre := Hz;
    b := buffer;
    ligbuf(b,l);
    ll := length(l);
    p := 1;
    prec := nil;
    num := 1;
    me.dc := nil ;(* ds le cas où il n'y a pas de libellé *)
    while p <= ll do
    begin
        while (p <= ll) and (l[p] = ' ') do p := p + 1 ;(* saute les blancs *)
        debut := p + me.hg.c -1;
        while (p <= ll) and (l[p]<>'.') and (l[p] <> '#' )do
               p := p + 1 ;(* Cherche fin *)
        if p <= ll
           then
            begin
              new(ch);
              if prec = nil then begin
                                   me.ch := ch;
                                   me.dc := ch;
                                 end
                            else begin
                                   prec^.suiv := ch ;
                                   ch^.prec := prec;
                                 end;
              ch^.de := debut;
              ch^.a  := p + me.hg.c - 2;
              ch^.num := num;
              num := num + 1;
              prec := ch;
              if l[p] = '#'
              then begin
                     p := p+ 1;
                     lirenuml(l,p,ch^.mesuiv);
                     me.genre := HzV;
                   end
              else begin
                     p := p + 1;
                     ch^.mesuiv := 0;
                   end;
            end;
    end;
    me.ch^.prec := ch;
    ch^.suiv := me.ch;

    bufmem(buffer,ll,me.pme);
```

```
    lect_fit(l,er);
end;

procedure vert(buffer : tampon ; var me : menu ;var l : ligne);
(*   OBJECTIF :  Mémorisation d'un menu ME de type vertical,  i.e. que
l'on se déplace dedans par un voyage  à la verticale.

     Entrées  :  BUFFER  :  L'espace mémoire où  l'on  doit stocker le
menu.
                 L : La première ligne du menu lue sur le fichier.

     Sorties  :  ME : le menu qui vient d'être lu et mémorisé.
                 L : La ligne du menu suivant.
*)

var p    : integer;     (* position courante dans la ligne *)
    ll   : integer ;    (* longueur de la ligne courante *)
    ch   : pch ;        (* pointeur vers le choix courant *)
    prec : pch ;        (* pointeur vers le choix précédent *)
    b     : tampon   ;  (* fin du buffer, 1ere place libre *)
    num  : integer ;    (* numéro du libellé courant *)
    ta : integer ;      (* taille de la fenêtre *)
    fin  : boolean;     (* détermine la fin d'un menu *)
    er   : integer;
begin
    b := buffer; prec := nil;
    me.genre := V;
    ll := length(l);
    if l[2] <> '═' then l[1] := ' '; (* faire disparaître le coin *)
    num := 1;
    fin := false;
    me.dc := nil ;(* ds le cas où il n'y a pas de libellé *)
    while not fin do
    begin
      if l[me_prem] = '└' then
      begin
          fin := true;
          if l[2] <> '═' then l[1] := ' ';
      end;
      ligbuf(b,l);
      if not fin then
      begin
        p := 1;
        while (p <= ll) and (l[p]<>'.') and (l[p] <> '#' )do p:= p + 1;
        (* Cherche fin *)
        if p <= ll then
        begin
          new(ch);
          if prec = nil then
          begin
            me.ch := ch;
            me.dc := ch;
          end
          else begin
            prec^.suiv := ch ;
            ch^.prec := prec;
          end;
          ch^.de := me.bd.l; ch^.a := 0 ;
          ch^.num := num;
          num := num + 1;
```

```
          prec := ch;
          if l[p] = '#'then
          begin
            p := p+ 1;
            lirenuml(l,p,ch^.mesuiv);
            me.genre := HzV;
          end
          else begin
            p := p + 1 ;
            ch^.mesuiv := 0;
          end;
        end;
      end;
      me.bd.l := me.bd.l + 1;
      lect_fit(l,er);
    end;
    me.bd.l := me.bd.l - 1;
    me.ch^.prec := ch;
    ch^.suiv := me.ch;
    bufmem(buffer,taille(me.hg,me.bd),me.pme);
end;

procedure lectaide(var l : ligne ; var me : menu);
(*   OBJECTIF         :  Associe  aux libéllés  leur  référence de menu
d'aide.

     Entrées          : L : la première ligne des aides.

     Sorties          : L : Une ligne après les lignes d'aide.
                        ME : Le menu auquel doit être associé les lignes
d'aide.
*)

var
    suiv : pch ; (* le libellé courant de la boucle *)
    er   : integer;
begin
    me.naide := l;(* aide associé au menu si choix = 0 *)
    if me.dc <> nil
    then begin
        suiv := me.dc;
        repeat    (* aide associé aux libellés *)
          lect_fit(l,er);
          suiv^.naide := l;
          suiv := suiv^.suiv;
       until suiv = me.dc;
   end;
   lect_fit(l,er);
end;

procedure lectcom(var l : ligne ; var me : menu);
(*   OBJECTIF         : Associe aux libéllés leur lignes de commentaire.

     Entrées          : L : la première ligne des commentaires.

     Sorties          : L : Une ligne après les lignes de commentaire.
                        ME : Le menu auquel doit être associé les lignes
de commentaire.
*)
```

```
var
    suiv : pch ; (* le libellé courant de la boucle *)
    comment : tampon;

begin
    if me.dc <> nil
    then begin
        suiv := me.dc;
        repeat
          commentaire(l,comment);
          suiv^.comm := comment;
          suiv := suiv^.suiv;
       until suiv = me.dc;
   end;
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 30                                                            I *)
(* I                          LECT DES MENUS                              I *)
(* I                                                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)

procedure lectmenus(buffer : tampon ; var t : tmenus);
(*   OBJECTIF       :  Création du tableau contenant tous les menus du
programme.

     Entrées        :  BUFFER :  Espace de travail pour la création du
tableau des menus.

     Préconditions  : BUFFER doit avoir été défini.

     Sorties        : T: le tableau contenant tous les menus.
*)


var ime : integer; (* indice de t *)
    l   : ligne ;  (* Ligne de caractères courante*)
    b   : tampon ;
    p   : integer ; (* indice de caractères dans une ligne *)
    er  : integer;
begin
    initme(buffer);

    lect_fit(l,er); (* la première ligne du fichier est le nom des écrans *)
    ime := 0;
    b := buffer;
    gotoxy(2,20); (* message de patience *)
    while l[me_prem] <> '.' do (* le fichier se termine par un ligne commençant*
)
    begin                          (*   par un point *)
      ime := ime + 1;
      write('¥');
      lect_fit(l,er);  (* ligne de position du menu *)
      p := 1;
      lirenuml(l,p,t[ime].hg.l);
      p := p + 1;
      lirenuml(l,p,t[ime].hg.c);
      if l[p] = '/' then
      begin
         p := p + 1;
         lirenuml(l,p,t[ime].lcomm);
```

```
      end
      else t[ime].lcomm := 0 ; (* pas de commentaire *)
      lect_fit(l,er);
      t[ime].bd.l := t[ime].hg.l;
      t[ime].bd.c := t[ime].hg.c + length(l) - 1;

      if l[me_prem] = '┌' then vert(b, t[ime],l)
                              else horiz(b, t[ime],l) ;
     t[ime].nme := ime;
     lectaide(l,t[ime]); (* association aux libellés de leur 'aide' *)
     if t[ime].lcomm <> 0
      then lectcom(l,t[ime]); (* association aux libellés de leur commentaire *)
    end;
  clotme(buffer);
end;
(*  >>>>>>>>>>>>>>>>>>>>>>>> VOYAGE DS MENUS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(* ┌-----------------------------------------------------------------------┐ *)
(* I NIVEAU 36                VOYAGE DS MENUS                               I *)
(* └-----------------------------------------------------------------------┘ *)
procedure AffLibHzCour(ou : point ; m : menu);
(*   OBJECTIF  : Affiche le libellé horizontal courant du menu ME à la
position OU.

     Entrées   :  OU  :  L'endroit  où  doit  être  affiché le libellé
courant.
                 M : le menu d'où provient le libellé.
*)

var i : integer;

begin
    gotoxy(ou.c,ou.l);
    write('                                        ');
    gotoxy(ou.c,ou.l);
    for i := m.dc^.de-m.hg.c to m.dc^.a-m.hg.c do
        write(chr(mem[seg(m.pme^):ofs(m.pme^)+i]));
end;

procedure nchoix_ref(me :menu ; nchoix : integer ;var  refch : pch );
(*   OBJECTIF : Fournit la référence du choix correspondant à un
numéro de choix.

     Entrée   : ME : Menu où se trouve le choix.
                     NCHOIX : Le numéro du choix.

     Sortie   : REFCH : La référence aux choix correspondant à NCHOIX.
*)

begin
    refch := me.ch;
    while (refch^.num <> nchoix) do refch := refch^.suiv;
end;

procedure aff_menu(me : menu);
(*   OBJECTIF         : Affiche le menu ME.

     Entrées          : ME : Le menu à afficher.
*)

var l,c,i : integer;
```

```
begin
  i := 0;
  for l := me.hg.l to me.bd.l do
    for c:= me.hg.c to me.bd.c do
    begin
      mem[adec : 2*(c+l*80-81)] := mem[seg(me.pme^):ofs(me.pme^)+i];
      i := i + 1;
    end;
end;

procedure libechgbd(me : menu ; var hg,bd : point);
(*   OBJECTIF : Détermine la position du libelle courant.

     Entrées  :  ME :  Le menu dans lequel il faut chercher le libellé
courant.

     Sorties  : HG, BD : Décrivent une fenêtre qui contient le libellé
courant.
*)

begin
   if me.genre = V
   then begin
      hg.l := me.dc^.de;
      hg.c := me.hg.c + 1;
      bd.l := hg.l;
      bd.c := me.bd.c - 1;
   end
   else begin
      hg.l := me.hg.l;
      hg.c := me.dc^.de;
      bd.l := hg.l;
      bd.c := me.dc^.a;
   end;
end;

procedure affcomm(me : menu);
(*   OBJECTIF : Affichage du commentaire associé au libellé courant.

     Entrée   : ME : Le menu qui contient le libellé courant.
*)

var col : integer;
    i : integer;
    ofse : integer;
    hg,bd : point;
begin
  if me.lcomm <> 0 then
  begin
    ofse := 2*(me.lcomm*80-80);(*adresse du 1er caractère de la ligne *)
    hg.l := me.lcomm;bd.l := me.lcomm;
    hg.c := 1 ; bd.c := 80 ;
    clsw(hg,bd);
    i := 1;
    for col := 1 to ord(me.dc^.comm^) do  (* longueur de la chaine *)
    begin
      mem[adec:2*col+ofse] := mem[seg(me.dc^.comm^):ofs(me.dc^.comm^)+i];
      i := i+1;
    end;
  end;
```

```
end;

(*  ┌──────────────────────────────────────────────────────────────────────┐ *)
(*  │ NIVEAU 34                VOYAGE DS MENUS                             │ *)
(*  └──────────────────────────────────────────────────────────────────────┘ *)
procedure menuw(me : menu);
(*   OBJECTIF   :  Affiche l'écran me et met dans la pile  ce qu'il va
écraser.

     Entrées    : ME : Le menu qui doit être affiché.
                  ST :  La pile de fenêtres où l'on empilera la partie
de l'écran écrasée.

     Sortie     :  ST : la pile contenant la partie écrasée.
*)

var hg,bd : point;
begin
(* efface la ligne d'un menu Hz *)
(* NEW *)
  if me.genre = HzV then
(* NEW *)
  begin
    hg := me.hg ; bd := me.bd;
    hg.c := 1 ; bd.c := 80;
    pushw(hg,bd,st);
    clsw(hg,bd);
  end
  else pushw(me.hg,me.bd,st);
  aff_menu(me);
  cachecurs;
end;


procedure initlib(me : menu);
(*   OBJECTIF       :  Affichage du libellé courant en inverse et
affiche la ligne de commentaire qui lui est associé.

     Entrées        :  ME  :  le menu duquel on extrait son libellé
courant.

     Postcondition  : Le libellé courant est affiché en inverse.
*)

var hg,bd : point;
begin
   libechgbd(me,hg,bd);
   invw(hg,bd);
   affcomm(me);
end;

procedure finlib(me : menu);
(*   OBJECTIF       :  Affichage du libellé courant en normal (non
inversé).

     Entrée         :  ME  :  le menu duquel on extrait son libellé
courant.

     Postcondition : Le libellé courant est affiché en normal.
*)
```

```
var hg,bd : point;
begin
   libechgbd(me,hg,bd);
   truew(hg,bd);
end;

procedure preclib(var me : menu);
(*   OBJECTIF :  Le libellé  courant  du  menu  ME  devient  celui qui
précède le courant.

     Entrée   : ME : le menu.
*)

var l,c,i : integer;
    hg,bd : point;
begin
   libechgbd(me,hg,bd);
   truew(hg,bd);
   me.dc := me.dc^.prec;
   libechgbd(me,hg,bd);
   invw(hg,bd);
  affcomm(me);
end;

procedure suivlib(var me : menu);
(*   OBJECTIF :  Le libellé courant du menu ME devient  celui qui suit
le courant.

     Entrée   : ME : le menu.
*)

var hg,bd : point;
begin
   libechgbd(me,hg,bd);
   truew(hg,bd);
   me.dc := me.dc^.suiv;
   libechgbd(me,hg,bd);
   invw(hg,bd);
  affcomm(me);
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 32                VOYAGE DS MENUS                             │ *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure premlib(var me : menu);
(*   OBJECTIF : Positionne le menu sur son premier libellé.

     Entrée   : ME : le menu.
*)

begin
    while (me.dc^.num <> 1) do suivlib(me);
end;

procedure voy_hz2(tc : touchectrl ; var me : menu  );
(* Utilisation :
               avant * menuw(me);
                       initlib(me);
                 + lecture des touches
```

```
                aprés * finlib(me);
                        popw(st);

    OBJECTIF  :  Voyage  dans  un  menu  horizontal  selon la méthode
décrite ci-dessus.

    Entrées   : ME : le menu courant.
                TC  :   la  touche  de  contrôle  utilisée  lors  du
déplacement.

    Sortie    : ME : le menu où le libellé courant est mis à jour.
*)

begin
     case tc of
       GAUCHE : if me.dc <> nil then preclib(me);
       DROITE : if me.dc <> nil then suivlib(me);
       F1     : if me.dc = nil then me_aide(me.naide)
                               else me_aide(me.dc^.naide);
     end;
end;

procedure voy_v2(tc:touchectrl ; var me : menu);
(* présuppose : avant * menuw(me);
                        initlib(me);
                   + lecture des touches
                aprés * finlib(me);
                        popw(st);

    OBJECTIF    : Voyage dans un menu vertical selon la méthode
décrite ci-dessus.

    Entrées     : ME : le menu courant.
                  TC  :   la  touche  de  contrôle  utilisée  lors  du
déplacement.

    Sorties     : ME : le menu où le libellé courant est mis à jour.
*)

begin
    case tc of
       HAUT    : if me.dc <> nil then preclib(me);
       BAS     : if me.dc <> nil then suivlib(me);
       F1      : if me.dc = nil then me_aide(me.naide)
                               else me_aide(me.dc^.naide);
    end;
end;
(* ┌──────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 31                VOYAGE DS MENUS                         │ *)
(* └──────────────────────────────────────────────────────────────────┘ *)
procedure voy_Hz( sctrl : sofctrl ;var me : menu ; var tc : touchectrl);
(*   OBJECTIF       :  Gestion  de  la  lecture  du  menu  ME  de type
horizontal.

    Entrées        :  SCTRL  :  l'ensemble des  touches  de contrôles
acceptables.
                      ME : Le menu sur lequel on travaille.

    Précondition   : Le menu est déjà affiché.
```

```
     Sortie          : TC : la dernière touche de contrôle utilisée.
*)

var    CH:char ;
begin
    initlib(me);
    repeat
      lect([], [gauche,droite,F1]+sctrl, CH, TC );
      voy_Hz2(tc,me);
    until tc in sctrl;
    finlib(me);
end;

procedure voy_V( sctrl : sofctrl ;var me : menu ; var tc : touchectrl);
(*   OBJECTIF        :  Gestion  de  la  lecture  du  menu  ME  de type
vertical.

     Entrées         :  SCTRL  :  l'ensemble  des  touches  de contrôle
acceptable.
                        ME : Le menu sur lequel on travaille.

     Précondition    : Le menu est déjà affiché.

     Sortie          : TC : la dernière touche de contrôle utilisée.
*)

var    CH:char ;
begin
    initlib(me);
    repeat
      lect([], [haut,bas,F1]+sctrl, CH, TC );
      voy_v2(tc,me);
    until tc in sctrl;
    finlib(me);
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 30                                                            I *)
(* I                          VOYAGE DANS LES MENUS                       I *)
(* I                                                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure voy_Hz0( sctrl : sofctrl ;var me : menu ; var tc : touchectrl);
(*   OBJECTIF        :  Affichage du menu Horizontal  et gestion totale
de sa lecture.

     Entrées         :  SCTRL  :  l'ensemble  des  touches  de contrôle
acceptable.
                        ME : Le menu sur lequel on travaille.

     Précondition    : Le menu n'est pas affiché.

     Sortie          : TC : la dernière touche de contrôle utilisée.

     Postcondition : A la sortie le menu a disparu de l'écran.
*)

begin
    menuw(me);
    voy_Hz( sctrl, me , tc);
    popw(st);
```

```
end;

procedure voy_V0( sctrl : sofctrl ;var me : menu ; var tc : touchectrl);
(*   OBJECTIF        :  Affichage du menu vertical ME et gestion totale
de sa lecture.

     Entrées         :  SCTRL  :  l'ensemble  des  touches  de contrôle
acceptable.
                        ME : Le menu sur lequel on travaille.

     Précondition    : Le menu n'est affiché.

     Sortie          : TC : la dernière touche de contrôle utilisée.

     Postcondition : A la sortie le menu a disparu de l'écran.
*)

begin
    menuw(me);
    voy_V( sctrl, me , tc);
    popw(st);
end;

procedure voy_HzV(sctrl : sofctrl ; var me : menu ; var tc : touchectrl);
(*   OBJECTIF        :  Affichage du menu ME où l'on peut se déplacer à
la fois  verticalement et  horizontalement  et  gestion  totale  de sa
lecture.

     Entrées         :  SCTRL  :  l'ensemble  des  touches  de contrôle
acceptable.
                        ME : Le menu sur lequel on travaille.

     Précondition    : Le menu n'est pas affiché.

     Sortie          : TC : la dernière touche de contrôle utilisée.

     Postcondition : A la sortie le menu a disparu de l'écran.
*)

var    CH:char ;
       meaide : menu;
begin
    initlib(me);
    if me.dc^.mesuiv <> 0 then
    begin
      menuw(tp[me.dc^.mesuiv]);
      initlib(tp[me.dc^.mesuiv]);
    end;
    repeat
      lect([], [gauche,droite,haut,bas,F1]+sctrl, CH, TC );
      case tc of
        GAUCHE : begin
                   if me.dc^.mesuiv <> 0 then
                   begin
                     finlib(tp[me.dc^.mesuiv]);
                     popw(st);
                   end;
                   preclib(me);
                   if me.dc^.mesuiv <> 0 then
                   begin
```

```
                      menuw(tp[me.dc^.mesuiv]);
                      initlib(tp[me.dc^.mesuiv]);
                   end;
                 end;
        DROITE : begin
                   if me.dc^.mesuiv <> 0 then
                   begin
                     finlib(tp[me.dc^.mesuiv]);
                     popw(st);
                   end;
                   suivlib(me);
                   if me.dc^.mesuiv <> 0 then
                   begin
                     menuw(tp[me.dc^.mesuiv]);
                     initlib(tp[me.dc^.mesuiv]);
                   end;
                 end;
          HAUT : begin
                   if me.dc^.mesuiv <> 0
                   then preclib(tp[me.dc^.mesuiv]);
                 end;
          BAS  : begin
                   if me.dc^.mesuiv <> 0
                   then suivlib(tp[me.dc^.mesuiv]);
                 end;
          F1   : begin
                   if me.dc^.mesuiv <> 0
                      then meaide := tp[me.dc^.mesuiv]
                      else meaide := me;
                   if meaide.dc = nil
                      then me_aide(meaide.naide)
                      else me_aide(meaide.dc^.naide);
                 end;
     end;
   until tc in sctrl;
   if me.dc^.mesuiv <> 0 then
   begin
     popw(st); (* le dernier menu vert *)
     finlib(tp[me.dc^.mesuiv]);
   end;
   finlib(me);
end;
(*   >>>>>>>>>>>>>>>>>>>>>>>>>>>>   ECRAN _ DEMO    <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(*  ┌───────────────────────────────────────────────────────────────────────────────────────┐ *)
(*  I NIVEAU 32                     ECRAN _ DEMO                                             I *)
(*  └───────────────────────────────────────────────────────────────────────────────────────┘ *)
procedure bas_cons;
(*   OBJECTIF         :  Affichage  dans le bas de  l'écran :
patient ; service ; page
*)

var hg,bd : point;
begin
  hg.l := 21 ; hg.c := 1 ;
  bd.l := 21 ; bd.c := 80;
  pushw(hg,bd,st);
  clsw(hg,bd);
  gotoxy(1,21);
  invvideo;
  write(
```

```
'Hienne Cesar        |  Commun          | Caractéristiques |
 '
);
   hg.l := 21; hg.c := 65;
   gotoxy(hg.l,hg.c);
   aff_date(hg,premdate);
   trvideo;
   cachecurs;
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 30                                                            I *)
(* I                          ECRAN _ DEMO                                I *)
(* I                                                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure af_ec_demo(fichier : nomfichier);

(*   OBJECTIF          :  Affiche  l'écran demo  et  empile  ce  qui est
écrasé.

     Entrées           :   FICHIER  :  le  fichier  contenant  l'écran  à
afficher.
*)

var hg,bd : point;
    l : ligne;
    er : integer;
begin
  hg.l := 1 ; hg.c := 1 ; bd.l := 20 ; bd.c := 80;
  debut_fit(fichier,er);
  pushw(hg,bd,st);
  clsw(hg,bd);
  bordsimple(hg,bd);
  lect_fit(l,er); (* ligne de commentaire *)
  lect_fit(l,er);
  while l[me_prem] <> '.' do
  begin
      gotoxy(hg.c+2,hg.l+1);
      write(l);
      lect_fit(l,er);
      hg.l := hg.l + 1;
   end;
   fin_fit(er);
   cachecurs;
nd;
```

Contenu

Ce niveau contient actuellement toutes les procédures de lecture de la définition d'un champ. Il etait prévu initialement pour contenir toutes les autres procédures de lecture propres à chaque fonction.

Procédures et spécifications

```
(*-------------------------------------------------------------------------*)
(*                                                                         *)
(*          ***********      NIVEAU   20              ***********         *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)
(*  ┌──────────────────────────────────────────────────────────────────┐   *)
(*  ¦ NIVEAU 28              LECTURE DES CHAMPS                        ¦   *)
(*  └──────────────────────────────────────────────────────────────────┘   *)
procedure init_champ(var ch : champ);
(*   OBJECTIF       : Mise dans la variable CHAMP de toutes les
valeurs par défaut.

     Sortie         : CHAMP : Le champ rempli de toutes les valeurs par
défaut.
*)
begin
   with ch do
   begin
       nom := 'NOM';
       genre := statique;
       repetitif := true;
       commun := false;
       lignecom := 'lignecom';
       tip := entier;
       unite := 'UNITE';
(* entier *)
       intinf := 0;   intsup := maxint;          intdef := 0;
(* réél *)
       reelinf := 0;  reelsup := 0;              reeldef := 0;
(* chaine *)
       chdef := 'Chaine par défaut';
(* date *)
       dateinf := mindate   ; datesup := maxdate ;   datedef := datejour;
(* heure *)
       heureinf := minheure ; heuresup := maxheure ; heuredef := '120000';
(* code *)
       nomcode := 'nomcode';  quantite := false;   codedef := 1;
   end;
end;

procedure aff_on(ou : point ; on : boolean);
(*   OBJECTIF  : Affiche la valeur 'OUI' ou 'NON' à la position OU.

     Entrées   : OU : L'endroit où le message doit être affiché.
                 ON : Booléen déterminant le résultat.
                          True  : Affiché 'OUI'
                          False : Affiché 'NON'
*)
begin
  gotoxy(ou.c,ou.l);
```

```
  if on then write('OUI ')
        else write('NON');
  cachecurs;
end;

procedure st_ch_ins;
(*   OBJECTIF   :  Simulation  de  l'affichage de la liste  des champs
disponibles.
*)
var tc : touchectrl;
    hg,bd : point;
begin
    hg.l := 23 ; hg.c := 30 ; bd.l := hg.l ; bd.c := hg.c + 16;
    invw(hg,bd);
    voy_V0([F1,return,esc],tp[me_s_lch],tc);
    truew(hg,bd);
end;

(* ------------------------------------------------------------------- *)
(* | NIVEAU 26              LECTURE DES CHAMPS                       | *)
(* ------------------------------------------------------------------- *)
procedure ou_cham(m : menu ; choix : integer ; var ou : point);
(*   OBJECTIF   :  Détermine  pour  un  menu  où  commence  le libellé
correspondant au choix.

     Entrées    : M : Le menu de référence.
                  CHOIX : Le choix courant dans ce menu.

     Sortie     : OU : Le point où commence le libellé courant.
*)
var refsh : pch;
begin
    ou.c := m.bd.c + 1;
    nchoix_ref(m,choix,refsh);
    ou.l := refsh^.de;
end;

procedure lc_unite(ou : point ; var cham : champ ; var tc : touchectrl ;
                   naide : refaide);
(*   OBJECTIF   : Lecture de l'unité du champ.

     Entrées    : OU : Endroit où doit s'effectuer la lecture.
                  NAIDE  :  Le numéro de l'écran d'aide  à afficher si
l'utilisateur utilise la touche F1.

     Sorties    :  CHAM :  Le  champ  contient  désormais  la nouvelle
unité.
                   TC :  La dernière  touche de contrôle  utilisée par
l'utilisateur.
*)
var c : char;
   result : chainec;
begin
    result := cham.unite;
    tc := F10;
    c := '';
    lc_chaine(ou,lunite,[' '..'z'],[F1,Ins,haut,bas,esc,return],c,tc,result,naid
e);
    cachecurs;
    cham.unite := result;
end;
```


Annexe Niv-2 2

```
(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 24              LECTURE DES CHAMPS                            │ *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
(* ########################  INITIALISATION  ############################## *)
procedure init_aff_champ(var cham : champ);
(*   OBJECTIF         : Initialisation de l'affichage pour le champ
CHAM.
     Ceci pour les parties communes à tous les champs, i.e. :
      Type / Nom / Commentaire / Commun / Genre / Répétitif.

     Entrées          : CHAM : Le champ à afficher.
*)
var refch : pch;
    ou : point;
begin
    ou.c := tp[mes_ch_en].bd.c + 1;

(* type *)
    while tp[mesc_tip].dc^.num <> (ord(cham.tip) + 1) do
       suivlib(tp[mesc_tip]);
    AffLibHzCour(tp[mesc_tip].hg,tp[mesc_tip]);

(* nom *)
    nchoix_ref(tp[mes_ch_en],2,refch);
    ou.l := refch^.de;
    gotoxy(ou.c,ou.l); write(cham.nom);

(* commentaire *)
    nchoix_ref(tp[mes_ch_en],3,refch);
    ou.l := refch^.de;
    gotoxy(ou.c,ou.l); write(cham.lignecom);

(* commun *)
    nchoix_ref(tp[mes_ch_en],4,refch);
    ou.l := refch^.de;
    aff_on(ou,cham.commun);

(* genre *)
    while tp[mesc_genre].dc^.num <> (ord(cham.genre) + 1) do
       suivlib(tp[mesc_genre]);
    AffLibHzCour(tp[mesc_genre].hg,tp[mesc_genre]);

(* répétitif *)
    nchoix_ref(tp[mes_ch_en],6,refch);
    ou.l := refch^.de;
    aff_on(ou,cham.repetitif);

end;

procedure InAfChEnt(var cham : champ);
(*   OBJECTIF    : Initialisation de l'affichage pour le champ  CHAM de
type entier.

     Entrée      : CHAM : Le champ vu sous l'angle entier à afficher.
*)

var ou : point;
begin
    init_aff_champ(cham);
```

```
    ou_cham(tp[mes_ch_en],7,ou);
    gotoxy(ou.c,ou.l); write(cham.intinf);
    ou_cham(tp[mes_ch_en],8,ou);
    gotoxy(ou.c,ou.l); write(cham.intsup);
    ou_cham(tp[mes_ch_en],9,ou);
    gotoxy(ou.c,ou.l); write(cham.intdef);
    ou_cham(tp[mes_ch_en],10,ou);
    gotoxy(ou.c,ou.l); write(cham.unite);
end;

procedure InAfChReel(var cham : champ);
(*   OBJECTIF    : Initialisation de l'affichage pour le  champ CHAM de
type réel.

     Entrée      : CHAM : Le champ vu sous l'angle Réel à afficher.
*)
var ou : point;
begin
    init_aff_champ(cham);
    ou_cham(tp[mes_ch_en],7,ou);
    ec_reel(ou,cham.reelinf);
    ou_cham(tp[mes_ch_en],8,ou);
    ec_reel(ou,cham.reelsup);
    ou_cham(tp[mes_ch_en],9,ou);
    ec_reel(ou,cham.reeldef);
    ou_cham(tp[mes_ch_en],10,ou);
    gotoxy(ou.c,ou.l); write(cham.unite);
end;

procedure InAfChCH(var cham : champ);
(*   OBJECTIF    : Initialisation de l'affichage pour le champ  CHAM de
type chaîne de caractères.

     Entrée      : CHAM : Le champ vu sous l'angle Chaîne de caractères
à afficher.
*)
var ou : point;
begin
    init_aff_champ(cham);
    ou_cham(tp[mes_ch_ch],7,ou);
    ou.l := ou.l + 2;
    ou.c := 2;
    gotoxy(ou.c,ou.l);
    write(cham.chdef);
end;

procedure InAfChDate(var cham : champ);
(*   OBJECTIF    : Initialisation de l'affichage pour le champ  CHAM de
type date.

     Entrée      : CHAM : Le champ vu sous l'angle Date à afficher.
*)

var ou : point;
begin
    init_aff_champ(cham);
    ou_cham(tp[mes_ch_da],7,ou);
    aff_date(ou,cham.dateinf);
    ou_cham(tp[mes_ch_da],8,ou);
    aff_date(ou,cham.datesup);
```

```
    ou_cham(tp[mes_ch_da],9,ou);
    aff_date(ou,cham.datedef);
end;

procedure InAfChHe(var cham : champ);
(*   OBJECTIF    : Initialisation de l'affichage pour le  champ CHAM de
type Heure.

     Entrée      : CHAM : Le champ vu sous l'angle Heure à afficher.
*)

var ou : point;
begin
    init_aff_champ(cham);
    ou_cham(tp[mes_ch_he],7,ou);
    aff_heure(ou,cham.heureinf);
    ou_cham(tp[mes_ch_he],8,ou);
    aff_heure(ou,cham.heuresup);
    ou_cham(tp[mes_ch_he],9,ou);
    aff_heure(ou,cham.heuredef);
end;


(* ##########################     COMMUN      ############################### *)
procedure lc_tip(var cham : champ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture du type du champ courant.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ dont le type est mis à jour.
*)
begin
    voy_hz0([F1,Ins,return,esc,haut,bas],tp[mesc_tip],tc);
    cham.tip := typ(tp[mesc_tip].dc^.num-1);
end;

procedure lc_nom(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture du type du nom du champ.

     Entrée      : M : Menu utilisé pour la lecture du nom.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ dont le nom est mis à jour.
*)

var c : char;
   result : chainec;
   ou : point;
begin
    result := cham.nom;
    tc := F10;
    c := '';
    ou_cham(m,2,ou);
    lc_chaine(ou,lnomch,[' '..'z'],[F1,Ins,haut,bas,return,esc],c,tc,result,
              m.dc^.naide);
    cachecurs;
    cham.nom  := result;
end;
```

```
procedure lc_lignecom(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture de ligne de commentaire  du champ courant.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ dont la ligne de commentaire est mis
à jour.
*)

var c : char;
   result : chainec;
   ou : point;
begin
    result := cham.lignecom;
    tc := F10;
    c := '';
    ou_cham(m,3,ou);
    lc_chaine(ou,llignecom,[' '..'z'],[F1,Ins,haut,bas,return,esc],c,tc,result,
              m.dc^.naide);
    cachecurs;
    cham.lignecom := result;
end;

procedure lc_commun(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture du caractère commun du champ courant.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM :  Le champ dont le caractère commun  est mis à
jour.
*)
var ou : point;
begin
    ou_cham(m,4,ou);
    lc_ouinon(ou,[F1,Ins,haut,bas,return,esc],cham.commun,tc,m.dc^.naide);
end;

procedure lc_genre(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture du genre du champ courant.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ dont le genre est mis à jour.
*)

begin
    voy_hz0([F1,Ins,return,esc,haut,bas],tp[mesc_genre],tc);
    cham.genre := genr(tp[mesc_genre].dc^.num-1);
    AffLibHzCour(tp[mesc_genre].hg,tp[mesc_genre]);
end;

procedure lc_repetitif(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture du caractère répétitif du champ courant.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
```


Annexe Niv 2.6

```
                CHAM :  Le champ dont le caractère commun  est mis à
jour.
*)

var ou : point;
begin
    ou_cham(m,6,ou);
    lc_ouinon(ou,[F1,Ins,haut,bas,return,esc],cham.commun,tc,m.dc^.naide);
end;

(* ##########################     ENTIER     ############################### *)
procedure lc_intinf(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF   :  Lecture de la borne inférieure du champ courant, vu
sous l'angle entier.

     Entrée     : M : Menu utilisé pour la lecture du type.

     Sortie     : TC : la dernière touche de contrôle frappée.
                  CHAM : Le champ mis à jour.
*)

var c : char;
    ou : point;
begin
    ou_cham(m,7,ou);
    lc_entier(ou,0,cham.intdef,[F1,Ins,haut,bas,return,esc],c,tc,cham.intinf,m.d
c^.naide);
end;

procedure lc_intsup(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF   :  Lecture de la borne supérieure du champ courant, vu
sous l'angle entier.

     Entrée     : M : Menu utilisé pour la lecture du type.

     Sortie     : TC : la dernière touche de contrôle frappée.
                  CHAM : Le champ mis à jour.
*)

var c : char;
    ou : point;
begin
    ou_cham(m,8,ou);
    lc_entier(ou,cham.intdef,maxint,[F1,Ins,haut,bas,return,esc],c,tc,
              cham.intsup,m.dc^.naide);
end;

procedure lc_intdef(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF   :  Lecture de la définition du champ  courant, vu sous
l'angle entier.

     Entrée     : M : Menu utilisé pour la lecture du type.

     Sortie     : TC : la dernière touche de contrôle frappée.
                  CHAM : Le champ mis à jour.
*)

var c : char;
    ou : point;
begin
    ou_cham(m,9,ou);
```


Annexe Niv-2 7

```
    lc_entier(ou,cham.intinf,cham.intsup,[F1,Ins,haut,bas,return,esc],c,tc,
              cham.intdef,m.dc^.naide);
end;

procedure lc_intunite(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture de l'unité du champ courant, vu sous l'angle
entier.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)

var ou : point;
begin
    ou_cham(m,10,ou);
    lc_unite(ou,cham,tc,m.dc^.naide);
end;

(* ##########################     REEL        ############################### *)
procedure lc_reelinf(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la borne inférieure du champ courant, vu
sous l'angle réel.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    ou : point;
begin
    ou_cham(m,7,ou);
    lc_reel(ou,0,cham.reeldef,[F1,Ins,haut,bas,return,esc],c,tc,
            cham.reelinf,m.dc^.naide);
end;

procedure lc_reelsup(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la borne supérieure du champ courant, vu
sous l'angle réel.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    ou : point;
begin
    ou_cham(m,8,ou);
    lc_reel(ou,cham.reeldef,maxreel,[F1,Ins,haut,bas,return,esc],c,tc,
              cham.reelsup,m.dc^.naide);
end;

procedure lc_reeldef(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la définition du champ courant,  vu sous
l'angle réel.

     Entrée      : M : Menu utilisé pour la lecture du type.
```

```
     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    ou : point;
begin
    ou_cham(m,9,ou);
    lc_reel(ou,cham.reelinf,cham.reelsup,[F1,Ins,haut,bas,return,esc],c,tc,
                cham.reeldef,m.dc^.naide);
end;

procedure lc_reelunite(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture de l'unité du champ courant, vu sous l'angle
réel.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var ou : point;
begin
    ou_cham(m,10,ou);
    lc_unite(ou,cham,tc,m.dc^.naide);
end;

(* ##########################     CHAINE      ############################### *)
procedure lc_chdef(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    : Lecture du contenu du champ courant, vu sous l'angle
chaîne de caractères.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    ou : point;
begin
    ou_cham(m,7,ou);
    ou.l := ou.l + 2;
    ou.c := 2;
    tc := F10;
    c := '';
    lc_chaine(ou,lchainec,[' '..'z'],[F1,Ins,haut,bas,return,esc],c,tc,
                cham.chdef,m.dc^.naide);
    cachecurs;
end;

(* ##########################     DATE        ############################### *)
procedure lc_dateinf(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la borne inférieure du champ courant, vu
sous l'angle date.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
```

```
var c : char;
    ou : point;
begin
    ou_cham(m,7,ou);
    lc_date(ou,mindate,cham.datedef,[F1,Ins,haut,bas,return,esc],c,tc,
            cham.dateinf,m.dc^.naide);
end;

procedure lc_datesup(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la borne supérieure du champ courant, vu
sous l'angle date.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    ou : point;
begin
    ou_cham(m,8,ou);
    lc_date(ou,cham.datedef,maxdate,[F1,Ins,haut,bas,return,esc],c,tc,
            cham.datesup,m.dc^.naide);
end;

procedure lc_datedef(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la définition du champ  courant, vu sous
l'angle date.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    ou : point;
begin
    ou_cham(m,9,ou);
    lc_date(ou,cham.dateinf,cham.datesup,[F1,Ins,haut,bas,return,esc],c,tc,
            cham.datedef,m.dc^.naide);
end;

(* ##########################      HEURE      ############################### *)
procedure lc_heinf(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la borne inférieure du champ courant, vu
sous l'angle heure.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)

var c : char;
    ou : point;
begin
    ou_cham(m,7,ou);
    lc_heure(ou,minheure,cham.heuredef,[F1,Ins,haut,bas,return,esc],c,tc,
            cham.heureinf,m.dc^.naide);
end;
```

```
procedure lc_hesup(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la borne supérieure du champ courant, vu
sous l'angle heure.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    ou : point;
begin
    ou_cham(m,8,ou);
    lc_heure(ou,cham.heuredef,maxheure,[F1,Ins,haut,bas,return,esc],c,tc,
              cham.heuresup,m.dc^.naide);
end;

procedure lc_hedef(var cham : champ ; var tc : touchectrl ; m : menu);
(*   OBJECTIF    :  Lecture de la définition du champ  courant, vu sous
l'angle heure.

     Entrée      : M : Menu utilisé pour la lecture du type.

     Sortie      : TC : la dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    ou : point;
begin
    ou_cham(m,9,ou);
    lc_heure(ou,cham.heureinf,cham.heuresup,[F1,Ins,haut,bas,return,esc],
            c,tc,cham.heuredef,m.dc^.naide);
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 22              LECTURE DES CHAMPS                            │ *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure lc_ch_en(hg,bd : point ; var cham : champ ; var tc : touchectrl);
(*   OBJECTIF    : Lecture du champ courant, vu sous l'angle ENTIER.

     Entrée      : HG, BD : points délimitant la fenêtre de lecture.

     Sortie      : TC : La dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    m : menu;
begin
    clsw(hg,bd);
    m := tp[mes_ch_en];
    aff_menu(m);
    InAfChEnt(cham);
    initlib(m);
    repeat
      voy_v2(tc,m);
      if tc = Ins then st_ch_ins;
      case m.dc^.num of
        1 : lc_tip(cham,tc,m);
        2 : lc_nom(cham , tc,m);
```

```
        3 : lc_lignecom(cham , tc,m);
        4 : lc_commun(cham , tc,m);
        5 : lc_genre(cham , tc,m);
        6 : lc_repetitif(cham , tc,m);
        7 : lc_intinf(cham , tc,m);
        8 : lc_intsup(cham , tc,m);
        9 : lc_intdef(cham , tc,m);
       10 : lc_intunite(cham , tc,m);
      end;
      if TC = F1 then TC := F10;
    until ((tc in [return,esc]) or (cham.tip <> entier));
    finlib(m);
end;

procedure lc_ch_re(hg,bd : point ; var cham : champ; var tc : touchectrl);
(*  (*   OBJECTIF    : Lecture du champ courant, vu sous l'angle REEL.

     Entrée      : HG, BD : Points délimitant la fenêtre de lecture.

     Sortie      : TC : La dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    m : menu;
begin
    clsw(hg,bd);
    m := tp[mes_ch_re];
    aff_menu(m);
    InAfChReel(cham);
    premlib(m);
    initlib(m);
    repeat
      voy_v2(tc,m);
      if tc = Ins then st_ch_ins;
      case m.dc^.num of
        1 : lc_tip(cham,tc,m);
        2 : lc_nom(cham , tc,m);
        3 : lc_lignecom(cham , tc,m);
        4 : lc_commun(cham , tc,m);
        5 : lc_genre(cham , tc,m);
        6 : lc_repetitif(cham , tc,m);
        7 : lc_reelinf(cham , tc,m);
        8 : lc_reelsup(cham , tc,m);
        9 : lc_reeldef(cham , tc,m);
       10 : lc_reelunite(cham , tc,m);
      end;
      if TC = F1 then TC := F10;
    until (tc in [return,esc]) or (cham.tip <> reel);
    finlib(m);
end;

procedure lc_ch_ch(hg,bd : point ; var cham : champ; var tc : touchectrl);
(*   OBJECTIF    :  Lecture du champ courant, vu sous l'angle CHAINE DE
CARACTERES.

     Entrée      : HG, BD : Points délimitant la fenêtre de lecture.

     Sortie      : TC : La dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
```

```
var c : char;
    m : menu;
begin
    m := tp[mes_ch_ch];
    clsw(hg,bd);
    aff_menu(m);
    InAfChCh(cham);
    premlib(m);
    initlib(m);
    repeat
      voy_v2(tc,m);
      if tc = Ins then st_ch_ins;
      case m.dc^.num of
        1 : lc_tip(cham,tc,m);
        2 : lc_nom(cham , tc,m);
        3 : lc_lignecom(cham , tc,m);
        4 : lc_commun(cham , tc,m);
        5 : lc_genre(cham , tc,m);
        6 : lc_repetitif(cham , tc,m);
        7 : lc_chdef(cham , tc,m);
      end;
      if TC = F1 then TC := F10;
    until (tc in [return,esc]) or (cham.tip <> chaine);
    finlib(m);
end;

procedure lc_ch_da(hg,bd : point ; var cham : champ ; var tc : touchectrl);
{*   OBJECTIF    : Lecture du champ courant, vu sous l'angle ENTIER.

     Entrée      : HG, BD : points délimitant la fenêtre de lecture.

     Sortie      : TC : La dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    m :menu;
begin
    m := tp[mes_ch_da];
    clsw(hg,bd);
    aff_menu(m);
    InAfChDate(cham);
    premlib(m);
    initlib(m);
    repeat
      voy_v2(tc,m);
      if tc = Ins then st_ch_ins;
      case m.dc^.num of
        1 : lc_tip(cham,tc,m);
        2 : lc_nom(cham , tc,m);
        3 : lc_lignecom(cham , tc,m);
        4 : lc_commun(cham , tc,m);
        5 : lc_genre(cham , tc,m);
        6 : lc_repetitif(cham , tc,m);
        7 : lc_dateinf(cham , tc,m);
        8 : lc_datesup(cham , tc,m);
        9 : lc_datedef(cham , tc,m);
      end;
      if TC = F1 then TC := F10;
    until (tc in [return,esc]) or (cham.tip <> dat);
    finlib(m);
```

```
end;

procedure lc_ch_he(hg,bd : point ; var cham : champ ; var tc : touchectrl);
(*   OBJECTIF    : Lecture du champ courant, vu sous l'angle HEURE.

     Entrée      : HG, BD : points délimitant la fenêtre de lecture.

     Sortie      : TC : La dernière touche de contrôle frappée.
                   CHAM : Le champ mis à jour.
*)
var c : char;
    m :menu;
begin
    m := tp[mes_ch_he];
    clsw(hg,bd);
    aff_menu(m);
    InAfChHe(cham);
    premlib(m);
    initlib(m);
    repeat
      voy_v2(tc,m);
      if tc = Ins then st_ch_ins;
      case m.dc^.num of
        1 : lc_tip(cham,tc,m);
        2 : lc_nom(cham , tc,m);
        3 : lc_lignecom(cham , tc,m);
        4 : lc_commun(cham , tc,m);
        5 : lc_genre(cham , tc,m);
        6 : lc_repetitif(cham , tc,m);
        7 : lc_heinf(cham , tc,m);
        8 : lc_hesup(cham , tc,m);
        9 : lc_hedef(cham , tc,m);
      end;
      if TC = F1 then TC := F10;
    until (tc in [return,esc]) or (cham.tip <> heur);
    finlib(m);
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 20              LECTURE DES CHAMPS                            │ *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure st_champ( var cham : champ);
(*   OBJECTIF    : Lecture du champ courant.

     Sortie      : CHAM : Le champ mis à jour.
*)
var
    tc : touchectrl;
    hg,bd : point;
begin
    hg.l := 1; hg.c := 1;
    bd.l := 23; bd.c := 80;
    pushw(hg,bd,st);
    bordsimple(hg,bd);
    gotoxy(30,23);
    write('Liste des Champs');
    hg.l := 2 ; hg.c := 2;
    bd.l := 22; bd.c := 79;
    repeat
    case cham.tip of
```

```
      entier : lc_ch_en(hg,bd,cham,tc);
      reel   : lc_ch_re(hg,bd,cham,tc);
      chaine : lc_ch_ch(hg,bd,cham,tc);
      dat    : lc_ch_da(hg,bd,cham,tc);
      heur   : lc_ch_he(hg,bd,cham,tc);
    end;
  until tc in [return,esc];
   gotoxy(30,23);
   write('                    ');
   popw(st);
end;
```

NIVEAU - 1

Contenu

C'est dans cette partie que se trouve la racine du programme, ainsi que les différentes rubriques et l'appel aux différentes fonctions.

Procédures et spécifications

```
program DOSHOPI;
{$i def.pas }
{$i n7.pas }
{$i n6.pas }
{$i n5.pas }
{$i n4.pas }
{$i n3.pas }
{$i n2.pas }

(*---------------------------------------------------------------------------*)
(*                                                                           *)
(*          ***********       NIVEAU   10          ***********               *)
(*                                                                           *)
(*---------------------------------------------------------------------------*)

procedure waitdemo(help : refaide);
(*   OBJECTIF          :  Permet  de  laisser  afficher  le  menu d'aide
courant tant que l'utilisateur tape F1.

     Entrées          : HELP : Le numéro du menu d'aide à afficher.
*)

var tc : touchectrl;
    c  : char;

begin
     lect([],[INS,F1,return,esc],c,tc);
     while tc = F1 do
     begin
        me_aide(help);
        lect([],[INS,F1,return,esc],c,tc);
     end;
end;

(*  >>>>>>>>>>>>>>>>>>>>>>>>>  INITIALISATION  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(* ┌-------------------------------------------------------------------┐ *)
(* I NIVEAU 12                INITIALISATION                           I *)
(* └-------------------------------------------------------------------┘ *)
procedure entete;
(*   OBJECTIF        : Affichage d'introduction au programme DOSHOPI.

     Postcondition : A la sortie l'affichage est en textmode(1)
*)

begin
 textmode(1);
 clrscr;
 gotoxy(1,3);
 writeln(' DAUPHIN présente');
 gotoxy(1,12);
 writeln('-┬┐  ┌┐  ┌─ │ │ ┌┐  ┌┐  │');
```

```
 writeln('  ');
 writeln('');
 writeln('');
 writeln('');
end;

PROCEDURE motdepasse;
(*   OBJECTIF       : Simulation de la lecture du nom du service et du
mot de passe correspondant.
*)

begin
  clrscr;
  textmode(1);
  writeln('Nom Du Service :');
  wait;
  writeln;
  writeln('Mot de Passe :');
  wait;
  textmode(2);
end;


procedure bureau;
(*   OBJECTIF       :  Affichage  sur tout l'écran  du caractère ASCII
177
*)

var i : integer;
    adres : tampon;
    hg, bd : point ;
begin
    for i := 0 to tecran - 3 * 80 - 1 do
        mem[adec:2*i] := 177;
end;
(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 10                                                            I *)
(* I                          INITIALISATION                              I *)
(* I                                                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure init;
(*   OBJECTIF       : Affichage du message d'introduction, lecture des
écrans constituant les menus,  lecture du mot de passe,  lecture de la
date du jour et insertion de cette dernière dans le  menu principal et
affichage de la table de travail.
*)

begin
    entete;
    initw(st);
    lectmenus(buffer,tp);
    motdepasse;
    clrscr;
    initdate;
    datemenu1;
    bureau;
end;

(*  >>>>>>>>>>>>>>>>>>>>>>>>  CONSULTATION    <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(*  ┌──────────────────────────────────────────────────────────────────────┐ *)
```

```
(* I NIVEAU 16                    CONSULTATION                                I *)
(* L-------------------------------------------------------------------------J *)
procedure cons_pat(var tc : touchectrl);
(*   OBJECTIF   :  Lecture  du  choix de la fonction  de consultation.
En fait la rubrique consultation ne contient qu'une seule fonction.

     Entrée     :  TC :  La dernière  touche  de  contrôle tapée avant
l'entrée dans cette procédure.

     Sortie     : TC : La dernière touche de contrôle frappée avant la
sortie de cette procédure.
*)
begin
   voy_V0([return,esc,gauche,droite],tp[me_cs_pat],tc);
end;

procedure cons_ser(var tc : touchectrl);
(*   OBJECTIF     :   Simulation  de  la  consultation   des  services
existants.

     Entrée     :  TC :  La dernière  touche  de  contrôle tapée avant
l'entrée dans cette procédure.

     Sortie     : TC : La dernière touche de contrôle frappée avant la
sortie de cette procédure.
*)
begin
   voy_V0([return,esc,gauche,droite],tp[me_cs_ser],tc);
end;

procedure cons_pag(var tc : touchectrl);
(*   OBJECTIF  : Simulation de la consultation des pages existantes.

     Entrée     :  TC  :  La dernière  touche  de  contrôle tapée avant
l'entrée dans cette procédure.

     Sortie     :  TC : La dernière touche de contrôle frappée avant la
sortie de cette procédure.
*)
begin
   voy_V0([return,esc,gauche,droite],tp[me_cs_pag],tc);
end;

procedure cons_dat(var tc : touchectrl);
(*   OBJECTIF  : Simulation du choix d'une date.

     Entrée     :  TC :  La dernière  touche  de  contrôle  tapée avant
l'entrée dans cette procédure.

     Sortie     :  TC : la dernière touche de contrôle frappée avant la
sortie de cette procédure.
*)
var me : menu;
    ou : point;
    c : char;
    naide : refaide;

begin
   naide := '75';
   ou.l := 21; ou.c := 65;
```

```
    trvideo;
    aff_date(ou,premdate);
    tc := F10;
    while not (tc in [return,esc,gauche,droite]) do
    begin
      lc_date(ou,mindate,maxdate,[gauche,droite,return],c,tc,premdate,naide);
    end;
    invvideo;
    aff_date(ou,premdate);
    trvideo;
    cachecurs;
end;

(*  ┌──────────────────────────────────────────────────────────────────────┐  *)
(*  │ NIVEAU 14                  CONSULTATION                              │  *)
(*  └──────────────────────────────────────────────────────────────────────┘  *)
procedure cons_choix(var me : menu ;var tc : touchectrl);
(*   OBJECTIF :  Lecture  du  type  d'argument  que l'utilisateur veut
modifier ou consulter.

    Entrées  :  TC  :  La dernière  touche  de  contrôle  tapée avant
l'entrée dans cette procédure.
                ME : Le menu contenant les différents choix possibles
ainsi que le choix courant.

    Sorties  :  TC :  La dernière touche de contrôle frappée avant la
sortie de cette procédure.
                ME : Le menu contenant les différents choix possibles
ainsi que le nouveau choix courant.
*)
begin
   case me.dc^.num of
     1 : cons_pat(tc);
     2 : cons_ser(tc);
     3 : cons_pag(tc);
     4 : cons_dat(tc);
   end;
end;

(*  ┌──────────────────────────────────────────────────────────────────────┐  *)
(*  │ NIVEAU 12                  CONSULTATION                              │  *)
(*  └──────────────────────────────────────────────────────────────────────┘  *)
procedure voy_cons( var me : menu);
(*   OBJECTIF :  Voyage  entre les différents  choix  possibles  de la
fabrique/Fonction consultation.

    Entrée   :  ME : le menu contenant les différentes possiblités de
la fonction consultation.

    Sortie   :  ME  :  le même menu où  le libéllé courant  a pu être
modifié.
*)
var ch :char;
    tc : touchectrl;

begin
   initlib(me);
   cons_choix(me,tc);
   while not (tc in [return,esc]) do
   begin
```

```
      voy_hz2(tc,me);
      cons_choix(me,tc);
    end;
    finlib(me);
end;
(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 10                                                            I *)
(* I                          CONSULTATION                                I *)
(* I                                                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure consultation;
(*   OBJECTIF       : Gestion des différentes options permises lors de
la consultation d'une page d'un dossier.
*)
var c : char;
    tc :touchectrl;
    hg, bd  : point;
    me : menu;
begin
  me := tp[me_cons];
  af_ec_demo('consulta.dem');
  menuw(me);
  bas_cons;
  tc := F10 ;
  repeat
     lect([],[INS,F1,return,esc],c,tc);
     case tc of
     INS : voy_cons(me);
     F1  : me_aide(me.naide);
     end;
  until tc in [return,esc];
  popw(st);(* bas_cons *)
  popw(st);(* memu *)
  popw(st);(* ecran *)
end;

(*  >>>>>>>>>>>>>>>>>>>>>>>>  STRUCTURE       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 18                STRUCTURE                                   I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure s_pa_lch(var tc : touchectrl);
(*   OBJECTIF  :  Simulation de la consultation de la liste des champs
existants lors de la création d'une page.

     Entrée    :  TC :  La dernière  touche  de  contrôle  tapée avant
l'entrée dans cette procédure.

     Sortie    :  TC : La dernière touche de contrôle frappée avant la
sortie de cette procédure.
*)
begin
    voy_V0([return,esc,gauche,droite],tp[me_s_lch],tc);
end;

procedure s_pa_lser(var tc : touchectrl);
(*   OBJECTIF : Simulation de la consultation des services existants.

     Entrée    :  TC  :  La dernière  touche  de  contrôle  tapée avant
l'entrée dans cette procédure.
```

```
     Sortie    :  TC :  la dernière touche de contrôle frappée avant la
sortie de cette procédure.
*)
begin
    voy_V0([return,esc,gauche,droite],tp[me_s_lser],tc);
end;

procedure s_pa_lpg(var tc : touchectrl);
(*   OBJECTIF  : Simulation de la consultation des pages existantes.

     Entrée    :  TC  :  La dernière  touche  de  contrôle tapée avant
l'entrée dans cette procédure.

     Sortie    :  TC : La dernière touche de contrôle frappée avant la
sortie de cette procédure.
*)
begin
    voy_V0([return,esc,gauche,droite],tp[me_s_lpg],tc);
end;

(* ┌-------------------------------------------------------------------┐ *)
(* ¦ NIVEAU 17                STRUCTURE                                ¦ *)
(* └-------------------------------------------------------------------┘ *)
procedure s_pa_choix(var me : menu ;var tc : touchectrl);
(*   OBJECTIF  :  Lecture  du  type d'argument que  l'utilisateur veut
modifier ou consulter.

     Entrées   :  TC  :  La dernière  touche  de  contrôle tapée avant
l'entrée dans cette procédure.
                 ME  :   Le  menu  contenant   les  différents  choix
possibles ainsi que le choix courant.

     Sorties   :  TC : La dernière touche de contrôle frappée avant la
sortie de cette procédure.
                 ME  :   Le  menu  contenant  les   différents  choix
possibles ainsi que le nouveau choix courant.
*)
begin
    case me.dc^.num of
      1 : s_pa_lch(tc);
      2 : s_pa_lser(tc);
      3 : s_pa_lpg(tc);
    end;
end;

(* ┌-------------------------------------------------------------------┐ *)
(* ¦ NIVEAU 16                STRUCTURE                                ¦ *)
(* └-------------------------------------------------------------------┘ *)
procedure voy_s_page( var me : menu);
(*   OBJECTIF :  Voyage dans  les  libellés  du  menu  de  la fonction
structure page.

     Entrée    : ME : le menu de la structure page.

     Sortie    : ME :  Le  menu  où  le libellé courant  peut avoir été
modifié.
*)
var ch :char;
    tc : touchectrl;
```

```
begin
    initlib(me);
    s_pa_choix(me,tc);
    while not (tc in [return,esc]) do
    begin
      voy_hz2(tc,me);
      s_pa_choix(me,tc);
    end;
    finlib(me);
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 14                  STRUCTURE                                 │ *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure st_page(var tc : touchectrl);
(*   OBJECTIF : Simulation de la constitution d'une page de dossier.

     Entrée   : TC : La dernière touche de contrôle tapée avant
l'entrée dans cette procédure.

     Sortie   : TC : La dernière touche de contrôle frappée avant la
sortie de cette procédure.
*)
var c : char;
    hg, bd  : point;
    me : menu;

begin
  af_ec_demo('st_page.dem');
  me := tp[me_s_pg];
  menuw(me);
  tc := F10 ;
  repeat
     lect([],[INS,F1,return,esc],c,tc);
     case tc of
     INS : voy_s_page(me);
     F1  : me_aide(me.naide);
     end;
  until tc in [return,esc];
  popw(st);(* memu *)
  popw(st);(* ecran *)
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* │ NIVEAU 12                  STRUCTURE                                 │ *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure st_ar_pa;
(*   OBJECTIF       :  Simulation de la consultation de l'architecture
d'une page.
*)
var  tc : touchectrl;
begin
    tc := F10 ;
    st_page(tc);
end;

procedure st_ar_ch;
(*   OBJECTIF       :  Simulation de la consultation de l'architecture
d'un champ.
*)
```

```
var c : char;
    tc : touchectrl;
    cham : champ;
begin
    tc := F10 ;
    init_champ(cham);
    st_champ(cham);
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 10                                                            I *)
(* I                          STRUCTURE                                   I *)
(* I                                                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure structure( choix : integer);
(*   OBJECTIF          :  Branchement  sur le  type  de  fonction  de la
rubrique structure.

     Entrées           : CHOIX : Le numéro de la fonction choisie.
*)
begin
  case choix of
    1 : st_ar_pa;
    2 : st_ar_ch;
    3 : st_ar_co;
  end;
end;

(*  >>>>>>>>>>>>>>>>>>>>>>>>  IMPRESSION      <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 12                IMPRESSION                                  I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure imp_lpat;
(*   OBJECTIF          :  Simulation de  l'impression  de  la  liste des
patients.
*)
begin
    af_ec_demo('imp_lpat.dem');
    waitdemo('994');
    popw(st);
end;

procedure imp_lser;
(*   OBJECTIF          :  Simulation de  l'impression  de  la  liste des
services.
*)
begin
    af_ec_demo('imp_lser.dem');
    waitdemo('994');
    popw(st);
end;

procedure imp_S_lpage;
(*   OBJECTIF          : Simulation de l'impression de la structure d'une
page.
*)
begin
    af_ec_demo('imp_S_lp.dem');
    waitdemo('994');
    popw(st);
```

```
end;


procedure imp_S_lchamp;
(*   OBJECTIF        :  Simulation de l'impression de la structure d'un
champ.
*)
begin
    af_ec_demo('imp_S_lc.dem');
    waitdemo('994');
    popw(st);
end;

procedure lect_lppage;
(*   OBJECTIF        :  Simulation de  la lecture de la  longueur d'une
page imprimante.
*)

var llpage : integer;
begin
    gotoxy(1,25);
    write('longueur d''une page d''impression ?  72');
    waitdemo('41');
    gotoxy(1,25);
    write('                                          ');
    cachecurs;
end;

(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 10                                                            I *)
(* I                          IMPRESSION                                  I *)
(* I                                                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure impression(choix : integer);
(*   OBJECTIF        : Simulation de la rubrique impression.

     Entrées         : CHOIX : Le numéro de la fonction sélectionnée.

     Sorties         : CHOIX : Le numéro de la fonction sélectionnée.
*)
begin
  case choix of
    1 : lect_lppage;
    2 : imp_lpat;
    3 : imp_lser;
    4 : imp_S_lpage;
    5 : imp_S_lchamp;
  end;
end;

(*  >>>>>>>>>>>>>>>>>>>>>>>>  AUTRES          <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(* ┌──────────────────────────────────────────────────────────────────────┐ *)
(* I NIVEAU 12                AUTRES                                      I *)
(* └──────────────────────────────────────────────────────────────────────┘ *)
procedure stockage;
(*   OBJECTIF        : Simulation de la fonction de stockage.
*)
var c : char;
    tc :touchectrl;
    hg, bd  : point;
```

```
    me : menu;
begin
    af_ec_demo('stockag1.dem');
    waitdemo('51');
    popw(st);
    af_ec_demo('stockag2.dem');
    waitdemo('51');
    popw(st);
end;

procedure reprise;
(*   OBJECTIF        : Simulation de la fonction de reprise des
informations.
*)
begin
    af_ec_demo('reprise1.dem');
    waitdemo('52');
    popw(st);
    af_ec_demo('reprise2.dem');
    waitdemo('52');
    popw(st);
end;

procedure m_mdp;
(*   OBJECTIF        : Simulation de la fonction de modification du mot
de passe.
*)
begin
    af_ec_demo('modmdp1.dem');
    waitdemo('53');
    popw(st);
    af_ec_demo('modmdp2.dem');
    waitdemo('53');
    waitdemo('53');
    popw(st);
end;

procedure ajout_pat;
(*   OBJECTIF        : Simulation de la fonction qui permet de rajouter
des patients à la liste de patients.
*)
var tc : touchectrl;
begin
    af_ec_demo('imp_lpat.dem');
    waitdemo('72');
    popw(st);
end;

procedure ajout_serv;
(*   OBJECTIF        : Simulation de la fonction qui permet de rajouter
des services à la liste des services.
*)
var tc : touchectrl;
begin
    af_ec_demo('imp_lser.dem');
    waitdemo('72');
    popw(st);
end;

(* ┌------------------------------------------------------------------┐ *)
```

```
(* I NIVEAU 10                                                          I *)
(* I                          AUTRES                                    I *)
(* I                                                                    I *)
(* L--------------------------------------------------------------------J *)
procedure autres(choix : integer);
(*   OBJECTIF : Simulation du fonctionnement de la rubrique AUTRES.

     Entrée   : CHOIX : Le numéro de la fonction sélectionnée.

     Sortie   : CHOIX : Numéro de la dernière fonction sélectionnée.
*)
begin
  case choix of
    1 : stockage;
    2 : reprise;
    3 : m_mdp;
    4 : ajout_pat;
    5 : ajout_serv;
   end;
end;

(*  >>>>>>>>>>>>>>>>>>>>>>>>  AUREVOIR       <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
(* ,--------------------------------------------------------------------, *)
(* I NIVEAU 10                                                          I *)
(* I                          AUREVOIR                                  I *)
(* I                                                                    I *)
(* L--------------------------------------------------------------------J *)
procedure aurevoir(tc : touchectrl ; var final : boolean);
(*   OBJECTIF : Simulation de la rubrique AUREVOIR.

     Entrée   : TC : la dernière touche de contrôle frappée.

     Sortie   :  FINAL : Booléen indiquant si le programme est terminé
ou non.
*)
begin
    if tc = RETURN then final := true;
end;

(*--------------------------------------------------------------------------*)
(*                                                                          *)
(*          ***********      NIVEAU   00              ***********          *)
(*                                                                          *)
(*--------------------------------------------------------------------------*)

(*  >>>>>>>>>>>>>>>>>>>>>>>>  PROGRAMME      <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< *)
procedure prgm;
(*   OBJECTIF        :   Programme   principal   dirigeant   vers   les
différentes rubriques.
*)
var final   : boolean;
    nmev : integer;  (* numéro du menu vertical *)
    tc : touchectrl;
begin
    init;
    final := false;
    repeat
      menuw(tp[me_prem]);
      voy_HzV([return],tp[me_prem],tc);
      popw(st);
```

```
      nmev := tp[me_prem].dc^.mesuiv;
      case tp[me_prem].dc^.num of
      1 : consultation;
      2 : structure(tp[nmev].dc^.num);
      3 : impression(tp[nmev].dc^.num);
      4 : autres(tp[nmev].dc^.num);
      5 : aurevoir(tc,final);
      end;
    until final;
end;


begin
   vers := hard;    (* Type de matériel Disque Dur ou Floppy. *)
   ecran_ext := 'dat';
   prgm;
end.
```

LES ECRANS

Contenu

Nous trouvons ici le contenu du fichier 'ECRAN.DAT' qui contient l'ensemble des écrans lus à l'initialisation du programme. La syntaxe utilisée pour ce fichier est explicitée en fin du fichier.

L'ensemble des écrans du programme

```
(* menu n° 1 *)
1,1/24   (* écran principal *)
Contenu#2     Structure#3    Impression#4   Divers#5        Sortie#6   jj/mm/aaaa
1
11
1
1
1
14
Création, Consultation, Modification ou Suppression du contenu d'une page.


Salut et à la prochaine !
(* menu n° 2 *)
2,1  (*  EX Contenu *)
```

Du dossier médical

```
11
(* menu n° 3 *)
2,15/24  (* Structure *)
```

. Page . Champ . Code

```
1
31
32
33
Création, Consultation, Modification ou Suppression de la structure d'une page
Création, Consultation, Modification ou Suppression de la définition d'un champ
Création, Consultation, Modification ou Suppression d'un code
(* menu n° 4 *)
2,30/24  (* Impression *)
```

.Longueur d'une page d'imprimante
.Liste des patients .Liste des services
.La liste des pages .La liste des champs

```
1
41
42
43
44
```

```
45
La longueur d'une page de l'imprimante.
Impression de la liste des patients
Impression de la liste des services
Impression de la liste des pages
Impression de la liste des champs
(* menu n° 5 *)
2,45/24  (* Autres  *)
```

```
. Archivage
. Reprise
```

```
   Modification
. Mot de passe
. Liste des patients
. Liste des services
```

```
1
51
52
53
54
55
Archivage de la base de données du service
Reprise de données à partir de l'archivage ou à partir d'une source extérieure.
Modification du mot de passe.
Modification de la liste des patients.
Modification de la liste des services.
(* menu n° 6 *)
2,60 (* EX Fin *)
```

```
  Fin du programme
```

```
14
(* écran n° 7 *)
22,1  (* Ins de Contenu *)
Nom du patient      .  Nom du service . Nom de la page    . jj/mm/aa ‹--› jj/mm/aa
.
11
72
72
72
75
(* écran n° 8 *)
6,1 (* Les patients *)
```

```
 Les patients existants :
```

```
.Aplon Phil
.Bambelle Larry
.Boquet Bill
.Coptere Elie
.Culture Sylvie
.Dament Evy
.Elebat Leo
.Hassin Marc
.Orial Edith
.Septe Jack
```

```
1
72
```

```
72
72
72
72
72
72
72
72
72
(* écran n° 9 *)
6,23 (* Les Services *)
```

```
 Les Services existants :
.Admission
.Anesthésie
.Commun
.Laboratoire
```

```
1
72
72
72
72
(* écran n° 10 *)
6,35 (* Les pages *)
```

```
 Les pages existantes :
 . 1 Les caractéristiques du patient :
 . 2 Le résumé des précédentes visites :
 .   2.1 Admission :
 .   2.2 Sortie :
 .   2.3 Diagnostiques et complications :
```

```
1
72
72
72
72
72
(* écran n° 11 *)
23,20 (* La date ou intervalle *)
```

```
     jj/mm/aa     <-->     jj/mm/aa
```

```
75
(* écran n° 12 *)
22,1  (* Ins de Structure pour une page *)
  Liste des champs#13       Liste des accès#14      Liste des pages#10
120
72
72
72
(* écran n° 13 *)
6,2 (* La liste des champs *)
```

```
 Liste des champs disponibles
 ----------------------------
.  DATE DE NAISSANCE
.  DATE MESURE
.C CODE ICD9CM
.  IDENTIF  V24
.  NATIONALITE
.  POULS
```

```
1
72
72
72
72
72
72
(* écran n° 14 *)
6,10 (* La liste des accès *)
```

```
 Liste des
 SERVICES          DEB-CS       FIN-CS       DEB-MD       FIN-MD

.Admission       01/01/1980   01/01/2000   01/01/1980   01/01/1980
.Anesthésie      01/01/1980   01/01/2000   01/01/1980   01/01/1980
.Laboratoire     01/01/1980   01/01/2000   01/01/1980   01/01/1980
```

```
1
72
72
72
(* écran n° 15 *)
0,0

1
(* écran n° 16 *)
2,2 (* L'introduction de la définition d'un champ : nombre *)
@@@@@@@@@@@@@@@@@@@@@@@@@@@@
.Type                    :@
                          @
.Nom du champ            :@
                          @
.Ligne de commentaire    :@
                          @
.Commun                  :@
                          @
.Genre                   :@
                          @
.Répétitif               :@
                          @
.Nombre Minimum          :@
                          @
.Nombre Maximum          :@
                          @
.Valeur par défaut       :@
                          @
```

```
 .Unité                    :@
└@@@@@@@@@@@@@@@@@@@@@@@@@@@@
1
72
72
72
72
72
72
72
72
72
72
(* écran N°17 *)
2,2 (* Pour une chaîne de caractères *)
┌@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 .Type                     :@
                            @
 .Nom du champ             :@
                            @
 .Ligne de commentaire     :@
                            @
 .Commun                   :@
                            @
 .Genre                    :@
                            @
 .Répétitif                :@
                            @
 .Valeur par défaut        :@
└@@@@@@@@@@@@@@@@@@@@@@@@@@@@
1
72
72
72
72
72
72
72
(* écran N° 18 *)
2,2 (* Pour une date *)
┌@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 .Type                     :@
                            @
 .Nom du champ             :@
                            @
 .Ligne de commentaire     :@
                            @
 .Commun                   :@
                            @
 .Genre                    :@
                            @
 .Répétitif                :@
                            @
 .Date Minimum             :@
                            @
 .Date Maximum             :@
                            @
 .Date par défaut          :@
└@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
72
72
72
72
72
72
72
72
72
(* écran N° 19 *)
2,2 (* Pour une heure *)
┌@@@@@@@@@@@@@@@@@@@@@@@@@
 .Type                   :@
                          @
 .Nom du champ           :@
                          @
 .Ligne de commentaire   :@
                          @
 .Commun                 :@
                          @
 .Genre                  :@
                          @
 .Répétitif              :@
                          @
 .Heure Minimum          :@
                          @
 .Heure Maximum          :@
                          @
 .Heure par défaut       :@
└@@@@@@@@@@@@@@@@@@@@@@@@@
1
72
72
72
72
72
72
72
72
72
(* écran n° 20 *)
2,2 (* Pour un code *)
┌@@@@@@@@@@@@@@@@@@@@@@@@@
 .Type                   :@
                          @
 .Nom du champ           :@
                          @
 .Ligne de commentaire   :@
                          @
 .Commun                 :@
                          @
 .Genre                  :@
                          @
 .Répétitif              :@
                          @
 .Nom du code            :@
                          @
 .Une Quantité           :@
                          @
 .Code par défaut        :@
```

```
└@@@@@@@@@@@@@@@@@@@@@@@@@
1
72
72
72
72
72
72
72
72
72
(* écran n° 21 *)
3,27 (* choix d'un champ type *)
Entier. Réel. Suite de caractères. Date. Heure.
1
72
72
72
72
72
(* écran n° 22 *)
11,27 (* choix d'un champ genre *)
Statique. Dynamique.
1
72
72
.
```

Mode d'emploi :

La première ligne du fichier n'est pas lue, mais doit donc exister.

Un écran se compose d'une POSITION :

---

col,ligne : endroit où doit apparaître le menu, le tout doit être suivi par un blanc.

Ceci peut-être suivi de '/ligne' qui indique qu'après le menu suit des lignes de commentaire à associer aux libellés et qu'il faut les afficher à la ligne : 'ligne'.

Et d'un MENU :

Il existe deux types de menus: Les formats présentés sont stricts:

1)En ligne

exemple : Azerty. Qwerty. ...

Chaque libellé doit se terminer par un point.

La ligne menu doit se terminer par un point.

2)En Colonne

```
Azerty.
bonjour
Qwerty#32
```

Seules les lignes contenant un point pourront être choisies.

Il est possible d'associer à un libellé un numéro de menu, dans ce cas on met dans la ligne où il apparaît # suivi du numéro du menu.

Pour faire disparaître le cadre :

```
┌@@@@@@@@@@@@@  Marquer le coin supérieur droit et inférieur droit.
@   Azerty. @  La  première ligne doit être  composée de '┌'  suivi de
@  bonjour  @      '@'
@  Qwerty#32@
└@@@@@@@@@@@@@  La dernière ligne : '└' suivi de '@'.
```

3)En ligne et en colonne :

Exemple :Azerty#1 Qwerty#2 ...

Chaque libellé se termine par un '#' suivi d'un numéro. le numéro désigne l'écran en colonne à afficher si l'on sélectionne le champ en question.

Remarque:

* Les points et '#' suivis de chiffres n'apparaissent pas à l'affichage.
* Chaque écran doit commencer par une ligne qui ne sera pas prise en considération.
* Chaque menu est suivi par un nom de fichier indiquant un fichier
  - Le premier pour le menu.
  - Les autres sont associés aux libéllés.
* A chaque libellé choisissable est associé une ligne de commentaire si la position du menu est suivi de '/numéro'.

* On termine les menus par un point en début de ligne. Cette ligne doit être accolée au dernier menu.

LES ECRANS D'AIDE ET DE DEMONSTRATION

1 Contenu

Nous trouvons ici l'ensemble des fichiers de simulation des écrans et les fichiers d'aide.

2 Fichier de démonstration

2.1 CONSULTA

CONSULTA.DEM

Caractéristiques :

Rue : Islakoul

N° : 12

Ville : Marignan

Code postal : 1515

Numéro national : 111.111.488.100

2.2 ICD9CM

ICD9CM.DEM

CODE : ICD9CM

...

754.8 Anomalie musculo-squelettique non tératogène, congénitale,autres.
754.81 Thorax en entonnoir congénital.
754.82 Thorax en carène, congénital.
754.89 Anomalie musculo-squelettique non tératogène, congénitale, autre.
754.891 Anomalie de la paroi thoracique, congénitale.
754.892 Arthrogrypose multiple, congénitale.

...

2.3 IMP_CONT

IMP_CONT.DEM

IMPRESSION DU CONTENU D'UNE PAGE

2.4 IMP_LPAT

IMP_LPAT.DEM

```
Aplon Phil
Bambelle Larry
Boquet Bill
Coptere Elie
Culture Sylvie
Dament Evy
Elebat Leo
Hassin Marc
Orial Edith
Septe Jack
```

2.5 IMP_LSER

IMP_LSER

```
Admission
Anesthésie
Commun
Laboratoire
```

2.6 IMP_S_LC

imp_S_LC.DEM

```
  DATE DE NAISSANCE
  DATE MESURE
C CODE ICD9CM
  IDENTIF  V24
  NATIONALITE
  POULS
```

2.7 IMP_S_LP

IMP_S_LP

```
1 Les caractéristiques du patient :
2 Le résumé des précédentes visites :
  2.1 Admission :
  2.2 Sortie :
  2.3 Diagnostiques et complications :
```

2.8 MODMDP1

MODMDP1.DEM

L'ancien mot de passe :

2.9 MODMDP2

MODMDP2.DEM

L'ancien mot de passe :

Le nouveau mot de passe :

2.10 REPRISE1

REPRISE1.DEM

Voulez-vous insérer la première disquette.

Taper 'Enter' une fois la diquette prête.

2.11 REPRISE2

REPRISE2.DEM

Voulez-vous insérer la première disquette.

Taper 'Enter' une fois la diquette prête.

Disquette N° 2 : Intervalle : 00/05/68 - 00/07/68

2.12 STARCOST

STARCOST.DEM

```
.Nom du CODE   : .........
.Commentaire   : ..........................................
.Commun        : Oui
.Genre         : Statique
.Répétitif     : Oui
.Valeur par défaut     : ..................................
```

.

2.13 STARCOMA

STARCOMA.DEM

```
  Groupe Sanguin
C ICD9CM
  Réveil
```

.

2.14 STOCKAG1

STOCKAG1.DEM

```
     Archivage des dossiers sur l'intervalle de temps :
                                  00/05/68 - 00/07/68
```

.

2.15 STOCKAG2

STOCKAG2.DEM

```
     Archivage des dossiers sur l'intervalle de temps :
                                  00/05/68 - 00/07/68

      Voulez-vous retirer ces données O/N ?
```

.

2.16 ST_AR_CO

ST_AR_CO.DEM

Groupe Sanguin
Réveil

.

2.17 ST_PAGE

st_page.dem

Caractéristiques :

Rue :

N° :

Ville :

Code postal :

Numéro national :

.

3 Fichier d'aide

3.1 HELP1

HELP1.HLP
écran d'aide numéro 1
Cet écran ne peut apparaître !!!!!!!!!!!!
.

3.2 HELP11

HELP11.HLP

Création, modification, consultation et destruction du contenu

d'une page d'un dossier médical.

Les commandes :

Home : La première ligne de la page
End : La dernière ligne de la page
PgUp : Avancer d'un écran
PgDn : Reculer d'un écran
: Monter d'une ligne
: Descendre d'une ligne

Ins : Modifier le patient, le service, la page ou la date
.

3.3 HELP14

HELP14.HLP

Sauvetage des dernières modifications effectuées lors de cette session.
.

3.4 HELP31

HELP31.HLP

Création, modification, consultation et destruction de la

structure d'une page d'un dossier médical.

Les commandes :

Home : La première ligne de la page
End : La dernière ligne de la page
PgUp : Avancer d'un écran
PgDn : Reculer d'un écran
: Monter d'une ligne
: Descendre d'une ligne

Ins : Ajouter un champ, Modifier la liste des accès,
Changer de page.
.

3.5 HELP32

HELP32.HLP

Création, modification, consultation et destruction de la

structure d'un champ.

Les commandes :

: Monter d'une ligne
: Descendre d'une ligne

‹-, -› : Pour choisir une option si plusieurs sont disponibles.

Ins : Changer de champ.

.

3.6 HELP33

HELP33.HLP

Création, modification, consultation et destruction de la
structure d'un code.

Pour ce faire deux écrans se suivent :

Dans un premier temps une liste des codes existant apparaît, elle permet la création, la destruction ou le choix d'un code.

Dans un second temps il est possible de modifier le contenu du code sélectionner.

.

3.7 HELP41

HELP41.HLP

Vous avez la possibilité de modifier la longueur d'une page de listing.

Les valeurs les plus courantes sont 66 et 72.

.

3.8 HELP42

HELP42.HLP

Impression partielle ou totale de la liste des patients.

.

3.9 HELP43

HELP43.HLP

Impression partielle ou totale de la liste des services de l'hôpital.

.

3.10 HELP44

HELP44.HLP

Impression partielle ou totale de la liste des pages du service.

.

3.11 HELP45

HELP45.HLP

Impression partielle ou totale de la liste des champs du service.

.

3.12 HELP51

HELP51.HLP

Archivage des données sur un intervalle de temps.

3.13 HELP52

HELP52.HLP

Reprise de données archivées.

.

3.14 HELP53

HELP53.HLP

Modification du mot de passe.

.

3.15 HELP54

HELP54.HLP

Modification de la liste des patients.

.

3.16 HELP55

HELP55.HLP

Modification de la liste des services.

.

3.17 HELP72

HELP72.HLP

COMMANDES DE PAGINATION :

Home : Le début de la liste.
End : La fin de la liste.
PgUp : Avancer d'un écran.
PgDn : Reculer d'un écran.
: Monter dans la liste d'un élément.
: Descendre dans la liste d'un élément.
F2 : Pour avoir une trace sur papier.

Fin de sélection :
Enter : Rechercher la page pour les paramètres rentrés.
Esc : Pour abandonner les dernières sélections.
<- : Changer le paramètre de gauche.
-> : Changer le paramètre de droite.

- / - : pour voir ou nom des éléments marqués.
Espace : pour modifier le statut d'un élément.

.

3.18 HELP75

HELP75.HLP

Choix de la date ou d'un intervalle de temps.

Les commandes : Les champs ici possibles : jour, mois et année

0 - 9 : La valeur d'un champ.
Home : La date de la première visite du patient enregistrée.
End : La date de la dernière visite du patient enregistrée.
: La date qui précède la date courante
dans la la liste des visites.
: La date qui suit la date courante
dans la la liste des visites.

Fin de sélection :
Enter : Rechercher la page pour les paramètres rentrés.
Esc : Pour abandonner les dernières sélections.
‹- : Changer le paramètre de gauche
-› : Changer le paramètre de droite

.

3.19 HELP990

HELP990.HLP

COMMANDES DE PAGINATION :

Home : Le début de la liste.
End : La fin de la liste.
PgUp : Avancer d'un écran.
PgDn : Reculer d'un écran.
: Monter dans la liste d'un élément.
: Descendre dans la liste d'un élément.
F2 : Pour avoir une trace sur papier.

Fin de sélection :
Enter : Rechercher le code.
Esc : pour abandonner la sélection.

Ins / Del : Pour créer ou détruire un code.
+ / - : Pour voir ou non des éléments marqués.
Espace : Pour modifier le statut d'un élément.
.

3.20 HELP992

HELP992.HLP

COMMANDES DE PAGINATION :

Home : Le début de la liste.
End : La fin de la liste.
PgUp : Avancer d'un écran.
PgDn : Reculer d'un écran.
: Monter dans la liste d'un élément.
: Descendre dans la liste d'un élément.
F10 : Pour développer ou non une branche.
F2 : Pour avoir une trace sur papier.

Fin de sélection :
Enter : Acceptation des modifications effectuées.
Esc : Pour abandonner les dernières modifications.

Ins / Del : pour créer ou détruire un élément de code.

3.21 HELP994

HELP994.HLP

COMMANDES DE PAGINATION :

Home : Le début de la liste.
End : La fin de la liste.
PgUp : Avancer d'un écran.
PgDn : Reculer d'un écran.
: Monter dans la liste d'un élément.
: Descendre dans la liste d'un élément.
F2 : Pour avoir une trace sur papier.

\+ : Pour visionner les éléments marqués.
\- : Pour sortir du mode de visualisation.
F10 : Sélection ou désélection de tous les éléments de la liste.